COMMODORE

COMMODORE 16

Bedienungshandbuch

COMMODORE 16

Bedienungshandbuch

Commodore

* INHALTSVERZEICHNIS *

	INHALTSVERZEICHNIS	iii
1.	TEIL 1 ANSCHLUSS UND INBETRIEBNAHME	1
1.1	GRUNDAUSSTATTUNG	2
1.2	SCHALTER UND ANSCHLÜSSE	3
	Ein/Aus-Schalter	3
	Netzgeräte-Anschluß	4
	'Reset'-Knopf	4
	'Joystick'-Anschlüsse	4
	Peripherie-Anschluß	5
	Steckmodul-Anschluß	5
	Datassetten-Anschluß	6
	Video-Anschluß	6
	RF-Anschlußbuchse für das Fernseh/Antennen-Kabel	7
1.3	INBETRIEBNAHME	7
	Monitorbetrieb mit dem COMMODORE 16	9
	Inbetriebnahme des COMMODORE 16	9
1.4	KLEINE FEHLERSUCHE	10
1.5	PERIPHERIE	11
2.	TEIL 2 BEDIENUNG DER TASTATUR	13
2.1	DAS TASTENFELD	14
2.2	SONDER-TASTEN	16
	<Return>	16
	<Shift>	17
	<Run/Stop>	17
	CURSOR-Steuerung	18
	<Inst/Del>	18
	<Home/Clear>	19
	<Control>	20

	<C=> (COMMODORE - Taste)	21
	<Rvs/On> <Rvs/Off>	22
	<Flash/On> <Flash/Off>	23
	<Esc>	24
2.3	FARB-TASTEN	25
2.4	GRAFIK-TASTEN	26
2.5	FUNKTIONS-TASTEN	28
	Die 'Help'-Taste	30
3.	TEIL 3 BENUTZUNG DER SOFTWARE	31
3.1	EINLEITUNG	32
3.2	STECKMODULE (CARTRIDGES)	32
	Laden von Steckmodul-Programmen	32
3.3	KASSETTEN	34
	Laden von Kassetten-Programmen	34
	Laden von bestimmten Kassetten-Programmen	36
	Speichern von Programmen auf Kassette	37
3.4	DISKETTEN	39
	Laden von Disketten-Programmen	40
	Formatieren (Headern) einer Diskette	42
	Speichern von Programmen auf Diskette	44
	Inhaltsverzeichnis (Directory) der Diskette	45
4.	TEIL 4 ERSTE SCHRITTE	47
4.1	EINLEITUNG	48
4.2	BILDSCHIRM	48
4.3	REVERSE DARSTELLUNG UND FARBÄNDERUNG	49
4.4	ERSTE PROGRAMMIERSCHRITTE	52
4.5	BEFEHLS-EINGABE	55
4.6	FEHLERBEHANDLUNG	56
4.7	BILDSCHIRM UND ARBEITSSPEICHER LÖSCHEN	60
4.8	BILDSCHIRM-FENSTER	61
4.9	ESCAPE-TASTENBEFEHLE	62
5.	TEIL 5 ZAHLEN UND RECHENOPERATIONEN	65
5.1	EINLEITUNG	66

5.2	ZAHLEN UND GRUNDRECHENFUNKTIONEN	66
	Brüche und Dezimalstellen	67
	Die 'π'-Taste	68
	Wissenschaftliche Notation	68
5.3	AUSFÜHRUNG VON RECHENOPERATIONEN	69
	Vorrangordnung bei Berechnungen	71
5.4	WEITERE BEFEHLE ZUM BILDSCHIRMAUSDRUCK	72
5.5	VARIABLE	75
	Numerische Funktionen	77
	Selbstdefinierte Funktionen	78
6.	TEIL 6 GRAFIK UND FARBEN	79
6.1	GRAFIKZEICHEN	80
	Wie man Spielkarten entwirft	80
6.2	ZEICHENBEWEGUNG (TRICK)	84
6.3	STEUERUNG DER FARBEN	88
	Nummern der Bildschirmbereiche	89
	Farb-Nummern	89
	Grafik-Modus	91
6.4	HOCHAUFLÖSENDE GRAFIK	93
6.5	PUNKTE, LINIEN UND ÜBERSCHRIFTEN	95
	Die CHAR-Anweisung	97
6.6	QUADRATE, KREISE, VIELECKE UND MALEREI	98
	Rechtecke zeichnen	98
	Die BOX-Anweisung	98
	Kreise zeichnen	100
	Die CIRCLE-Anweisung	101
	Die PAINT-Anweisung	103
6.7	MEHRFARBIGE GRAFIK	103
7.	TEIL 7 TÖNE UND MUSIK	107
7.1	EINLEITUNG	108
7.2	LAUTSTÄRKEREGELUNG (VOL)	109
7.3	TONERZEUGUNG (SOUND)	109
7.4	KLANGEFFEKTE ERZEUGEN	111
	Wir machen Musik	114
	COMMODORE 16 - die Musikmaschine	116

BASIC 3.5 LEXIKON 119

EINLEITUNG 120

KOMMANDO- UND ANWEISUNGSFORMAT 122

KOMMANDOS IN BASIC 3.5 125
- AUTO 125
- BACKUP 125
- COLLECT 126
- CONT 126
- COPY 127
- DELETE 128
- DIRECTORY 128
- DLOAD 129
- DSAVE 130
- HEADER 130
- HELP 131
- KEY 131
- LIST 133
- LOAD 134
- NEW 135
- RENAME 136
- RENUMBER 136
- RUN 137
- SAVE 137
- SCRATCH 139
- VERIFY 139

ANWEISUNGEN IN BASIC 3.5 141
- BOX 141
- CHAR 142
- CIRCLE 143
- CLOSE 144
- CLR 144
- CMD 144
- COLOR 145
- DATA 146
- DEF FN 146
- DIM 147
- DO/LOOP/WHILE/UNTIL/EXIT 148
- DRAW 149
- END 149
- FOR...TO...STEP 150
- GET- 151
- GETKEY 152
- GET# 153
- GOSUB 153
- GOTO 154
- GRAPHIC 154
- IF...THEN (...ELSE) 155
- INPUT 157
- INPUT# 158
- LET 158

LOCATE 159
MONITOR 159
NEXT 160
ON 161
OPEN 161
PAINT 163
POKE 164
PRINT 165
PRINT# 166
PRINT USING 167
PUDEF 172
READ 173
REM 173
RESTORE 174
RESUME 174
RETURN 175
SCALE 175
SCNCLR 176
SOUND 176
SSHAPE/GSHAPE 177
STOP 178
SYS 179
TRAP 179
TRON 180
TROFF 180
VOL 180
WAIT 181

WEITERE INFORMATIONEN ZU DEN GRAPHIK-ANWEISUNGEN . . 182

FUNKTIONEN 183
Numerische Funktionen 183
ABS(X) 183
ASC(X) 183
ATN(X) 183
COS(X) 184
DEC(H$) 184
EXP(X) 184
FNxx(X) 184
INSTR 184
INT(X) 185
JOY(N) 185
LOG(X) 185
PEEK(X) 186
RCLR(N) 186
RDOT(N) 186
RGR(X) 186
RLUM(N) 186
RND(X) 186
SGN(X) 188
SIN(X) 188
SQR(X) 188
TAN(X) 188
USR(X) 188
VAL(X$) 189

String-Funktionen . . . 191
CHR$(X) . . . 191
ERR$(N) . . . 191
HEX$(N) . . . 191
LEFT$(X$,X) . . . 191
LEN(X$) . . . 191
MID$(X$,S,X) . . . 192
RIGHT$(X$,X) . . . 192
STR$(X) . . . 192

Sonstige Funktionen . . . 193
FRE(X) . . . 193
POS(X) . . . 193
SPC(X) . . . 193
TAB(X) . . . 193
π . . . 193

VARIABLEN UND OPERATOREN . . . 195

VARIABLEN . . . 195
Gleitkomma-Variablen . . . 195
Ganzzahl-Variablen . . . 195
String-Variablen . . . 195

VARIABLEN-NAMEN . . . 196

MATRIZEN / FELDER . . . 196

RESERVIERTE VARIABLEN-NAMEN . . . 197
ST . . . 197
TI TI$ . . . 198
DS DS$ . . . 198
ER EL ERR$ . . . 198

BASIC-OPERATOREN . . . 199
Arithmetische Operatoren . . . 199
Vergleichsoperatoren . . . 200
Logische Operatoren . . . 200

ANHANG . . . 201

BEFEHLS-ABKÜRZUNGEN . . . 202

FEHLERMELDUNGEN . . . 204

DISK-FEHLERMELDUNGEN . . . 206

ABGELEITETE MATHEMATISCHE FUNKTIONEN . . . 210

NOTENWERTE . . . 211

KODE-TABELLEN . . . 212
Bildschirm-Kode . . . 212
ASC- und CHR$-Kodes . . . 215
Bildschirm-Steuerzeichen . . . 217

MASCHINENSPRACHEN-MONITOR TEDMON 218
TEDMON-Befehlssatz 218
Benutzen von TEDMON 219
Befehl A (Assemble) 219
Befehl C (Compare) 220
Befehl D (Disassemble) 220
Befehl F (Fill) 221
Befehl G (Go) 221
Befehl H (Hunt) 222
Befehl L (Load) 222
Befehl M (Memory) 223
Befehl > (nach M) 223
Befehl R (Register) 224
Befehl S (Save) 224
Befehl T (Transfer) 225
Befehl V (Verify) 225
Befehl X (Exit) 225

BITS UND BYTES 226

SPEICHERBELEGUNG 228

INDEX 229
A 229
B, C 230
D 231
E 232
F 233
G, H 234
I, J, K 235
L 236
M, N 237
O, P 238
Q, R 239
S 240
T, U 242
V, W, X, Y 243
Z 244

```
********************************
*                              *
*  T E I L   1                 *
*                              *
********************************
*                              *
*  A N S C H L U S S           *
*                              *
*  U N D                       *
*                              *
*  I N B E T R I E B N A H M E *
*                              *
********************************
```

*   GRUNDAUSSTATTUNG
     ****
*   SCHALTER UND ANSCHLÜSSE
     ****
*   INBETRIEBNAHME
     ****
*   KLEINE FEHLERSUCHE
     ****
*   PERIPHERIE
     ****

****
* 1.1 GRUNDAUSSTATTUNG *
****

Sie haben Ihren neuen COMMODORE 16 nun ausgepackt und fanden dabei auch dieses Handbuch. Prüfen Sie zuerst an Hand der nachfolgenden Liste, ob alle Teile der Grundausstattung vorhanden sind:

* Der COMMODORE 16
****

* Das Netzgerät
****

* Das Fernseh/Antennen-Kabel
****

* Das Bedienungshandbuch (in dem Sie bereits lesen!)
****

* Die Garantiekarte
****

Sollte Ihre Grundausstattung nicht vollständig sein, so klären Sie dies bitte sofort mit Ihrem COMMODORE-Händler.

Bevor Sie irgend einen Anschluß bzw. eine Verbindung an Ihrem Computer vornehmen, machen Sie sich mit seinen Schaltern und Verbindungsmöglichkeiten vertraut. Dann werden der Anschluß und die Inbetriebnahme schnell und ohne Schwierigkeiten möglich sein.

```
**********************************
*  1.2  SCHALTER UND ANSCHLÜSSE  *
**********************************
```

Beginnen wir mit der rechten Seite des COMMODORE 16

EIN/AUS SCHALTER
****

Der COMMODORE 16 muß stets AUS-geschaltet sein, wenn Sie:

* Steck-Module (Cartridges) einstecken oder entfernen
* Peripherie-Geräte wie z.B. Drucker, Datassette anschließen oder entfernen

Beachten Sie zur Kontrolle die mit 'POWER' beschriftete rote Lampe, die sich in der rechten, oberen Ecke der Tastatur befindet - sie leuchtet bei EIN-geschaltetem Computer rot!

NETZGERÄTE-ANSCHLUSS
****

In diese mit 'POWER' beschriftete Buchse wird das Niederspannungskabel des Netzgeräts gesteckt, während das zweite Kabel in die Netzsteckdose paßt.

'RESET'-KNOPF
****

Mit Hilfe des 'Reset'-Knopfes läßt sich der COMMODORE 16 neu starten - genau wie dies beim Einschaltvorgang geschieht. Dabei wird nicht nur der Bildschirm gelöscht, sondern auch jedes eingetippte BASIC-Programm! Wie Sie den Bildschirm löschen können, ohne dabei auch die Programme zu verlieren, wird im Teil 4 erklärt.

JOYSTICK-ANSCHLUESSE
****

An die mit 'JOY 1' und 'JOY 2' beschrifteten Anschlußbuchsen kann je ein 'Joystick' angeschlossen werden. Diese speziell für den COMMODORE 16 bestimmten 'Joysticks' erhalten Sie als Zubehör bei Ihrem COMMODORE-Händler.

Fahren wir mit der Rückseite des COMMODORE 16 fort:

PERIPHERIE-ANSCHLUSS
****

Weitere Peripheriegeräte - z.B. Disketten-Laufwerk, Drucker, u.a. - können über die mit der Aufschrift 'SERIAL' versehene Buchse angeschlossen werden.

Bei Verwendung mehrerer Peripheriegeräte beginnen Sie am COMMODORE 16 mit dem Anschluß des Disketten-Laufwerks (kurz 'Floppy' genannt) und schließen den Drucker mit einem weiteren Verbindungskabel am zweiten Floppy-Anschluß an.

STECKMODUL-ANSCHLUSS
****

Steckmodule (sog. 'Cartridges') bzw. Erweiterungen lassen sich über diesen mit 'MEMORY EXPANSION' bezeichneten Anschluß mit dem COMMODORE 16 verbinden.

Dabei ist stets darauf zu achten, daß sowohl <u>VOR</u> dem Einsatz bzw. auch <u>VOR</u> der Entnahme des Steckmoduls der COMMODORE 16 <u>AUS</u>-geschaltet wird!

DATASSETTEN-ANSCHLUSS
****

An dieser Anschlußbuchse wird die COMMODORE-'Datassette' 1531 angeschlossen.

VIDEO-ANSCHLUSS
****

Das beste Bild erzielen Sie über einen Farbmonitor, z.B. den COMMODORE 1701. Verwenden Sie hierzu ein Videokabel, das an der einen Seite einen 8-poligen Diodenstecker (DIN 41524) für die Video-Buchse Ihres COMMODORE 16 hat, auf der anderen Seite einen zu dem Monitor passenden Anschluß.

Falls Ihr Fernseher eine Video-Buchse hat, können Sie ihn ebenfalls als Monitor verwenden. Sie müssen nur dafür sorgen, daß der Video-Anschluß Ihres Fernsehers als Eingang geschaltet wird. Dies geschieht bei Verwendung eines Video-Recorders in der Regel durch eine Hilfsspannung von 12V, die an Pin 1 der Video-Buchse des Fernsehers gelegt wird.

Ihr Commodore 16 gibt eine solche Hilfsspannung nicht ab. Lassen Sie sich daher von Ihrem Fernsehfachmann eine geeignete Umschaltung einbauen. Bei manchen Fernsehern genügt dazu schon eine Drahtbrücke im Video-Stecker, da bei diesen an Pin 5 der Video-Buchse die benötigte Hilfsspannung von 12V bereitgestellt wird. Benutzen Sie daher diesen Eingang mit äußerster Vorsicht! Die Hilfsspannung darf auf keinen Fall an die Ausgänge Ihres COMMODORE 16 gelangen. Der Rechner würde dadurch sofort zerstört. Lassen Sie die Verbindung nur durch einen Fachmann herstellen!

RF-ANSCHLUSSBUCHSE FÜR DAS FERNSEH/ANTENNEN-KABEL
****

An dieser Stelle schließen Sie das eine Ende des mitgelieferten (dünnen) Fernseh/Antennen-Kabels an den Computer an. Das andere Kabelende wird in die Antennenbuchse Ihres Fernsehgeräts gesteckt.

****
* 1.3 INBETRIEBNAHME *
****

Die Zusammenschaltung des COMMODORE 16 mit den zugehörigen Komponenten ist denkbar einfach! Es sind nur drei Punkte, die zu beachten sind:

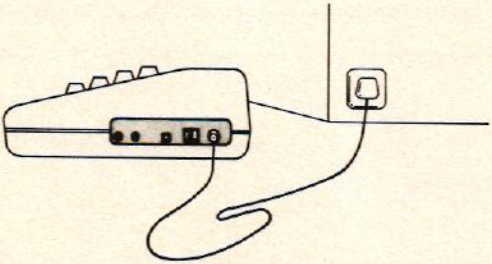
1. Sorgen Sie für mindestens zwei freie Netz-Steckdosen - eine für den COMMODORE 16 und eine für den Fernsehempfänger. Außer für die 'Datassette' muß für jedes weitere Peripheriegerät ebenfalls eine freie Steckdose zur Verfügung stehen.

2. Verbinden Sie das mitgelieferte Netzgerät mit Steckdose und Computer.

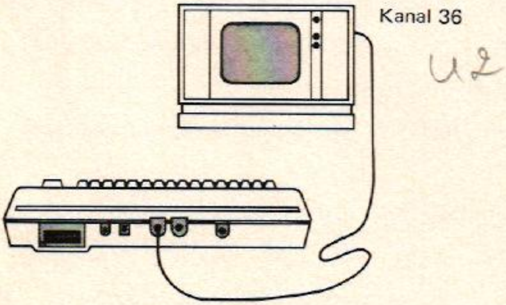
3. Verbinden Sie den Fernsehempfänger oder den Monitor mit dem COMMODORE 16.

Achten Sie unbedingt darauf, daß sämtliche Geräte AUS-geschaltet sind, solange nicht alle Verbindungen hergestellt wurden!

Zu 1: Es empfiehlt sich ein angemessener Abstand vom Computer zum Fernsehschirm (ca. 3-5 fache Bildschirm-Diagonale), sowie ausreichende Stellfläche für Arbeitsunterlagen und Peripherie.



Zu 2: Der kleine runde Stecker paßt in die entsprechende 'POWER'-Buchse am COMMODORE 16. Das Kabel mit dem Netzstecker wird in eine freie Netzsteckdose gesteckt.



Zu 3: Für die Zeit der Computerbenutzung müssen Sie das reguläre Antennenkabel am Fernsehgerät abziehen und statt dessen das Antennen/Verbindungskabel des COMMODORE 16 in den Antennen-Eingang des Fernsehgerätes stecken. Das andere Kabelende verbinden Sie mit dem 'RF'-Ausgang des Computers. Den Fernseher stellen Sie bitte auf Kanal 36 ein.

MONITORBETRIEB MIT DEM COMMODORE 16

Verwenden Sie anstelle eines Fernsehempfängers einen Monitor, so beachten Sie die Bedienungsanleitung des Monitors. Der Anschluß des COMMMODORE Color-Monitors 1701 ist denkbar einfach:

Sie benötigen nur das dem Monitor beigefügte Kabel, um den Monitor-Eingang mit dem 'VIDEO'-Ausgang des COMMODORE 16 zu verbinden.

INBETRIEBNAHME DES COMMODORE 16

Wenn Sie bisher aufmerksam mitgelesen haben, so wissen Sie sicher noch, daß sich der EIN/AUS Schalter des COMMODORE 16 an der rechten Seite befindet, und können den Computer nun einschalten.

Auf dem Bildschirm muß nun folgende Meldung erscheinen:

```
COMMODORE BASIC V3.5 12277 BYTES FREE
READY.
■
```

CURSOR (blinkend)

Der blinkende Cursor unter der 'READY'-Meldung zeigt an, daß der COMMODORE 16 auf weitere Eingaben wartet. Die Hintergrundfarbe sollte dabei hellblau und die Schrift schwarz sein.

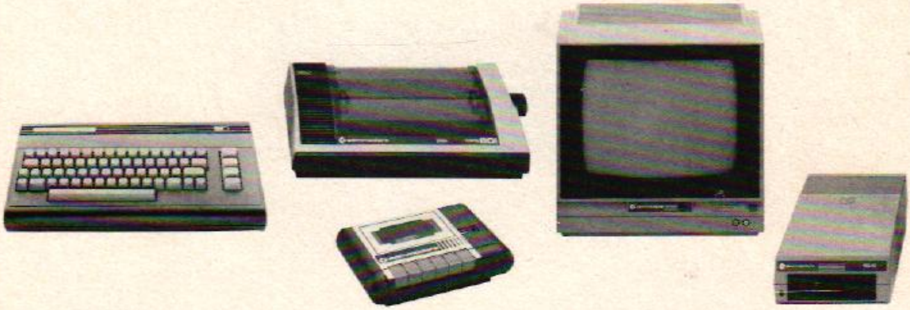
Wenn Probleme auftreten und sich der Computer nicht wie beschrieben meldet, so überprüfen Sie nochmals den Aufbau anhand nachfolgender Tabelle.

****
* 1.4 KLEINE FEHLERSUCHE *
****

SYMPTOM	URSACHE	MASSNAHME
Rote Kontroll-lampe brennt nicht	Computer nicht eingeschaltet	EIN/AUS Schalter muß auf 'ON'-Position stehen
	Keine Verbindung zum Netz	Netzstecker, Gerätestecker und Steckdose überprüfen
	Sicherung defekt	Sicherung durch Fach-händler auswechseln lassen
Keine Anzeige auf dem Bildschirm	Falscher Kanal am Fernsehgerät	Kanaleinstellung am Fern-sehgerät überprüfen
Zeichenwirrwarr auf dem Bildschirm bei eingestecktem Modul	Steckmodul nicht korrekt eingesteckt	Computer ausschalten und Sitz des Steckmoduls überprüfen
Schlechte Farben	Fernsehgerät nicht korrekt eingestellt	Farb- und Kanaleinstellung am Fernsehgerät prüfen
Starkes Rauschen	Lautstärke am Fern-sehgerät zu hoch eingestellt	Lautstärke verringern
Bild in Ordnung, aber kein Ton	Lautstärke am Fern-sehgerät zu niedrig eingestellt	Lautstärke erhöhen

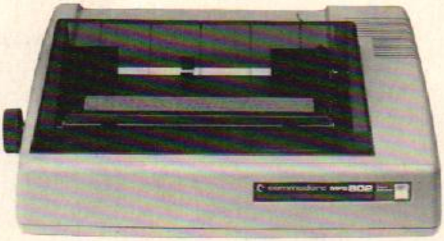
****
* 1.5 PERIPHERIE *
****

Unter Peripherie sind Erweiterungen Ihres COMMODORE 16 zu verstehen, mit denen Sie die Einsatzmöglichkeiten des Computers um ein Vielfaches vergrößern können. Dieses Zubehör erhalten Sie bei Ihrem COMMODORE-Händler. Mit diesen Peripheriegeräten lassen sich nicht nur Daten sichern und abspeichern, sondern z.B. auch alles, was auf dem Bildschirm erscheint, in Farbe oder in Schwarz-Weiß ausdrucken. Auch auf Kassette oder auf Diskette gespeicherte Programme können nun geladen werden und per Farb-Monitor noch besser als auf dem Fernsehgerät dargestellt werden.



Um Programme abzuspeichern oder wieder zu laden, wird ein externes Speichermedium benötigt. Dafür eignen sich sowohl Kassetten als auch Disketten. Beim Einsatz der Kassetten benötigen Sie die COMMODORE 'Datassette' 1531. Wollen Sie Disketten verwenden, stehen Ihnen zahlreiche Disketten-Laufwerke von COMMODORE zur Auswahl. Diese 'Floppies' sind nicht nur schnell, sondern auch sehr effizient.

Um mit Ihrem Computer ein Maximum an Bild- und Farbqualität zu erhalten, benötigen Sie statt des Fernsehempfängers einen COMMODORE Farb-Monitor.



Ob Sie mit einem Textverarbeitungs-Programm oder einem Grafik-Paket arbeiten, stets werden Sie zur Darstellung der Bildschirminhalte auf Papier einen Drucker benötigen. Wählen Sie zwischen Druckern, mit denen Sie Grafik und/oder Text 'plotten', oder sehr schnell ausdrucken, oder in Schönschriftqualität schreiben können. Ihr COMMODORE-Händler wird Sie gerne beraten.

```
*******************************
*                             *
*  T  E  I  L    2            *
*                             *
*******************************
*                             *
*  B  E  D  I  E  N  U  N  G  *
*                             *
*  D  E  R                    *
*                             *
*  T  A  S  T  A  T  U  R     *
*                             *
*******************************
```

- DAS TASTENFELD
- SONDER-TASTEN
- FARB-TASTEN
- GRAFIK-TASTEN
- FUNKTIONS-TASTEN
- DIE 'HELP'-TASTE

```
*************************
*  2.1  DAS TASTENFELD  *
*************************
```

Viele der Tasten des COMMODORE 16 gleichen denen einer Schreibmaschine. Beim COMMODORE 16 hat jedoch jede der Tasten - im Gegensatz zur Schreibmaschine - mehr als zwei Funktionen. In diesem Kapitel werden Sie lernen, mit Sonder-Tasten wie z.B. der COMMODORE-Taste <C=> oder auch den CURSOR-Steuertasten zu arbeiten. Sie werden die Funktionen jeder einzelnen Taste kennenlernen, eingeschlossen den Gebrauch der Grafiksymbole, wie sie auf der Vorderseite der Tasten aufgedruckt sind. Während der Erläuterungen sollten Sie sich mit der Lage jeder einzelnen Taste vertraut machen und ihren Gebrauch üben.

BENUTZEN SIE DIE TASTATUR WIE BEI EINER SCHREIBMASCHINE.

Wenn Sie die Buchstaben auf der Tastatur des COMMODORE 16 drücken, dann erscheinen diese als Großbuchstaben auf dem Bildschirm. Buchstaben und Ziffern sind dabei mit den auf den Tasten aufgemalten Zeichen identisch. Auch diverse andere Tasten, wie die meisten Interpunktionszeichen und Rechenoperationen, sind mit direktem Tastendruck erreichbar. Einige Interpunktionszeichen hingegen erreichen sie nur in Verbindung mit der Taste <Shift>.

Sie können den COMMODORE 16 auch in einen 'echten' Schreibmaschinen-Modus umschalten: Drücken Sie die Tasten <C=> und <Shift> gleichzeitig, und alle in der Folge gedrückten Buchstaben werden als Kleinbuchstaben auf dem Bildschirm erscheinen. In Verbindung mit der Taste <Shift> gedrückte Buchstaben erscheinen nun als Großbuchstaben. Ziffern und Interpunktionszeichen bleiben von dieser Modus-Umschaltung unbeeinflußt. Erneutes Drücken der Tastenkombination <C=> und <Shift> schaltet wieder in den Großbuchstaben-Modus zurück.

****
* 2.2 SONDER-TASTEN *
****

Einige Tasten des COMMODORE 16 sind auf einer herkömmlichen Schreibmaschine nicht zu finden. Mit Hilfe dieser Tasten sind die anderen Tasten mit zusätzlichen Funktionen belegt oder lassen sich bestimmte Aktionen des Computers steuern.

Darüberhinaus enthält das Tastenfeld des COMMODORE 16 etliche Sonderzeichen, die sowohl auf den meisten Schreibmaschinen- als auch Computertastaturen fehlen. Zu diesen Sonderzeichen zählt das 'Pfund-Sterling-Symbol' (£), das Pi (π), die Größer/Kleiner-Zeichen (> <), eckige Klammern ([]), Pfeilzeichen (←↑) und der 'Klammeraffe' (@).

<Return>
****

Die <u>Return</u>-Taste ist als Abschluß jeder Eingabezeile zu drücken, die Sie in den COMMODORE 16 über die Tastatur eingetippt haben. Erst mit Betätigung dieser Taste werden Daten und Befehle an den Computer gesendet. Der CURSOR springt immer an den Anfang der nächsten Zeile.

Wird die Return-Taste zusammen mit <Shift> gedrückt, springt der CURSOR zwar auch auf den nächsten Zeilenanfang, der Inhalt der alten Zeile wird jedoch <u>nicht</u> als Befehl (oder Programmzeile) an den Computer übergeben!

<Shift>
****

Ein Einsatzbeispiel mit der Taste <Shift> haben Sie bereits kennengelernt: die Tastatur des COMMODORE 16 als Schreibmaschinen-Tastatur. Dabei zeigte sich die typische Eigenheit der Taste <Shift>: sie wird stets benötigt, um andere Tasten bei der Bildschirmdarstellung zu unterstützen - allein gedrückt geschieht hingegen gar nichts. Die einrastende <Shift-Lock>-Taste bewirkt eine Dauerfunktion der <Shift>-Taste.

Mit Hilfe der Taste <Shift> erreichen Sie über die Tastatur nicht nur Großbuchstaben, sondern auch Grafik-Zeichen, Interpunktionszeichen und ein paar Dinge mehr - bei zusätzlicher Unterstützung durch einige weitere Tasten. Im Verlauf dieses Kapitels werden Sie noch mehr über die Taste <Shift> erfahren, wie man z.B. Grafikzeichen erzeugen kann.

<Run/Stop>
****

Um ein im COMMODORE 16 laufendes BASIC-Programm anzuhalten, drücken Sie diese Taste. Der Computer verläßt daraufhin das Programm und kehrt in den 'READY'-Modus zurück (Ausnahme: im INPUT-Befehl).

Werden die Tasten <Shift> und <Run/Stop> gleichzeitig gedrückt, so lädt der COMMODORE 16 das erste auf der eingelegten Diskette befindliche Programm und startet es in der Folge auch automatisch.

<CURSORSTEUERUNG>
****

Die vier Richtungspfeile in der obersten Tastenreihe geben die Bewegungsrichtung des blinkenden CURSORS an. Werden sie in eine Richtung gedrückt, so bewegt sich der CURSOR in diese Richtung. Es stehen vier Richtungen zur Verfügung: nach oben - unten - links - rechts. Während der Bewegung des CURSORS können, ohne Beeinflussung, vorhandene Bildschirmzeichen überlaufen werden. Bei Dauerdruck tritt, wie bei den anderen Tasten des COMMODORE 16, eine 'Repeat'-Funktion ein, d.h. das Zeichen oder die Funktion wird automatisch wiederholt, solange die Taste gedrückt bleibt. Damit ersparen Sie sich mühevolle Tastenwiederholungen.

<Inst/Del>
****

Mit Hilfe dieser Taste lassen sich Buchstaben, Ziffern und/oder Zeichen einfügen ('Insert') bzw. entfernen ('Delete'). Wenn Sie nur die Taste <Inst/Del> drücken, so wird bei jedem Tastendruck das links vom CURSOR befindliche Zeichen gelöscht und seine Stelle dafür vom CURSOR eingenommen. Auch kann damit ein Buchstabe aus der Zeile gelöscht werden: Sie bewegen den CURSOR einfach mit dem Tastenkreuz auf das dem zu entfernenden Buchstaben folgende Zeichen, drücken einmal die Taste <Inst/Del>, und der Buchstabe wird mit dem 'unter' dem CURSOR befindlichen Zeichen 'überschrieben'. Damit keine Lücke entsteht, verschiebt sich der Rest der Zeile ab der CURSOR-Position um eine Stelle nach links.

Wollen Sie in einer bestehenden Zeile Platz schaffen, um z.B. Buchstaben und/oder Zeichen einzufügen, dann gehen Sie mit dem CURSOR an die betreffende Stelle und betätigen die Taste <Inst/Del> gleichzeitig mit der Taste <Shift>. Der gewünschte Leerraum wird entsprechend der Anzahl Tastendrücke r e c h t s vom CURSOR bereitgestellt; der CURSOR selbst bewegt sich dabei nicht, sondern bleibt auf der Stelle. Mit dem entstehenden Leerraum rückt der gesamte Rest der Zeile nach rechts. Achtung: Sie sind jetzt bis zur Ausfüllung des erzeugten Leerraums im 'Quote-Modus', in dem Cursorbewegungen durch die Cursortasten nicht möglich sind!

Dank der Taste <Inst/Del> läßt sich beim Verbessern oder 'Editieren' von Texten, Programmlisten u.ä. viel Zeit sparen. Nochmals in der Zusammenfassung:

GEDRÜCKTE TASTE	plus +	TASTE	ergibt: =	
1. <Inst/Del>			=	löscht das vom CURSOR linksstehende Zeichen
2. <Inst/Del>	+	<Shift>	=	fügt an der CURSOR-Position eine Leerstelle ein

<Home/Clear>
****

Mit dieser Taste stehen Ihnen zwei Funktionen zur Verfügung. Drücken Sie nur die Taste <Home/Clear>, so springt der CURSOR in die linke obere Ecke, die sog. 'Home'-Position. Am gesamten Bildschirminhalt ändert sich dadurch nichts.

Drücken Sie hingegen die Taste <Shift> und gleichzeitig die Taste <Home/Clear>, springt der CURSOR ebenfalls in die 'Home'-Position, a b e r zusätzlich wird der gesamte Bildschirminhalt g e l ö s c h t! Zurück bleibt einzig der blinkende CURSOR in der linken, oberen Bildschirmecke.

Zum besseren Verständnis nochmals in der Zusammenfassung:

GEDRÜCKTE TASTE	plus +	TASTE	ergibt: =	
1. <Home/Clear>			=	bringt den CURSOR in die 'Home'-Position
2. <Home/Clear>	+	<Shift>	=	löscht den gesamten Bildschirminhalt und bringt den CURSOR in die 'Home'-Position

<Control>
****

Ebenso wie die Taste <Shift> funktioniert die Taste <Control> ausschließlich in Kombination mit anderen Tasten - ist also, allein gedrückt, wirkungslos. Sie muß gedrückt bleiben, während eine weitere Taste betätigt wird. Die Taste <Control> kommt in drei Fällen zur Anwendung:

1. Wie im Kapitel über die FARB-TASTEN noch näher erklärt wird, bewirkt das gleichzeitige Drücken der Taste <Control> und einer der für Farbe zuständigen Tasten (Zifferntasten <1> bis <8>) einen Farbwechsel der danach auf den Bildschirm geschriebenen Textzeilen.

2. Um ein Programm, das etwas auf Papier ausdruckt (PRINT) oder am Bildschirm gelistet (LIST) wird, anzuhalten, drücken Sie die Taste <Control> und gleichzeitig die Taste <S>. Mit Drücken irgendeiner Taste kann der Ausdruck (Print oder List) weiter fortgesetzt werden.

3. Die Taste <Control> wird auch gemeinsam mit den Tasten <Rvs/On> (ReVerSe = Negativ-Darstellung von Zeichen) bzw. <Rvs/Off> (Normal, Zifferntasten <9> bzw. <0>) und den Tasten <Flash/On> (FLASH = Blinken/Blitzen) bzw. <Flash Off> (Interpunktionstasten <,> bzw. <.>) benützt. Auch darüber mehr im Rest des Kapitels.

Ergänzend sei noch vermerkt, daß einige Software-Pakete die Taste <Control> mit eigenen, speziellen Funktionen belegt haben.

<C=> (COMMODORE-TASTE)
****

Die COMMODORE-Taste ist in ihrer Funktion der Taste <Control> sehr ähnlich und wird zur Unterstützung von insgesamt vier Funktionen (stets zusammen mit einer weiteren Taste) benötigt:

1. Wird die COMMODORE-Taste <C=> in Verbindung mit der Taste <Shift> betätigt, so wird die Tastatur in den Schreibmaschinen-Modus umgeschaltet (Groß- und Kleinschreibung). Auf die gleiche Weise kann man wieder in den Großschreib-Modus zurückschalten.

2. Die COMMODORE-Taste <C=> verhält sich außerdem wie eine weitere SHIFT-Taste, um die links über den Tasten aufgemalten Grafik-Symbole zu erreichen. COMMODORE-Taste <C=> und gleichzeitig die Taste mit dem betreffenden Grafik-Symbol drücken - und das gewünschte Zeichen steht auf dem Bildschirm.

3. Ebenfalls (wie die Taste <Control>) kann die COMMODORE-Taste <C=> benützt werden, um die auf den Schirm zu schreibende Textzeile in ihrer Farbe zu ändern. Nähere Angaben hierzu im nachfolgenden Kapitel.

4. Der Ablauf eines durchlaufenden (scrollenden) Bildschirm-Listings läßt sich durch Drücken der COMMODORE-Taste <C=> beträchtlich verlangsamen, so daß Programmzeile für Programmzeile gelesen werden kann. Wird der Tastendruck wieder aufgehoben, erfolgt das 'Scrollen' wieder mit normaler Geschwindigkeit.

<Rvs/On> und <Rvs/Off>
****

Mit dem COMMODORE 16 lassen sich die Bildschirmzeichen auch in 'REVERS'-Darstellung (Negativ-Bild) schreiben.

Blinkt der CURSOR z.B. in schwarzer Farbe auf gelb eingefärbtem Bildschirmhintergrund und betätigen Sie nun die Taste <Control> gleichzeitig mit der Taste <Rvs/On>, so erscheinen die danach eingegebenen Buchstaben und Zeichen in gelber Schrift auf schwarzem Hintergrund (also 'revers').

Dieser REVERS-Modus bleibt solange bestehen, bis Sie entweder die Taste <Control> gleichzeitig mit der Taste <Rvs/Off>, oder die Taste <Return>, oder zuerst die Taste <Esc> (Escape) und danach die Taste <O> drücken. Mit jeder dieser Tastenbedienungen kommen Sie zurück in den normalen (nicht-reversen) Schreibmodus.

	GEDRÜCKTE TASTE	plus +	TASTE	ergibt: =	
1.	<Control>	+	<Rvs/On>	=	Reverser Schreibmodus
2.	<Control>	+	<Rvs/Off>	=	Normaler Schreibmodus

<Flash/On> und <Flash/Off>

Durch diese Tasten können Zeichen auf dem Bildschirm zu ständigem Blinken (ähnlich dem CURSOR-Blinken) gebracht werden. Nachdem Sie die Taste <Control> und gleichzeitig die Taste <Flash/On> gedrückt haben, wird jedes geschriebene Zeichen ständig blinken.

Dieser BLINK-Modus bleibt solange bestehen, bis Sie entweder die Taste <Control> gleichzeitig mit der Taste <Flash/Off>, oder die Taste <Return>, oder die Taste <Esc> mit anschließendem 'O' drücken. Mit einer dieser Tastenbedienungen kommen Sie zurück in den normalen (nicht-blinkenden) Schreibmodus.

	GEDRÜCKTE TASTE	plus +	TASTE	ergibt: =	
1.	<Control>	+	<Flash/On>	=	Blinkender Schreibmodus
2.	<Control>	+	<Flash/Off>	=	Normaler Schreibmodus

<Esc> (Escape)
****

Durch die Taste <Esc> werden eine Vielzahl spezieller Bildschirm-Editierfunktionen möglich, einschließlich der Befehle für die Einrichtung von Bildschirmfenstern und ihrer Bedienung. Die Möglichkeit, Bildschirm-Fenster einzurichten, ist eine besondere Fähigkeit des COMMODORE 16. Sie sind damit in der Lage, Teile des Bildschirmes für andere Arbeiten zu nützen, ohne dabei bestehende Bildschirminhalte zu beeinflussen.

Mit der Taste <Esc> können nicht nur eine Vielzahl von Fenster-Editierbefehlen erreicht werden, sondern auch Funktionen wie Einfügen, Löschen sowie Scrollen werden möglich. Alle diese Möglichkeiten der Taste <Esc> werden im Rahmen des Kapitels 'Fenster-Techniken' (im Teil 4) besprochen.

****
* 2.3 FARB-TASTEN *
****

Die Farbtasten stehen in engem Verhältnis zu den Zifferntasten <1> bis <8>. Egal, ob eine dieser Tasten gemeinsam mit der Taste <Control> oder der COMMODORE-Taste <C=> gedrückt wird, stets verändern sie in der Folge die Farbe des danach geschriebenen Bildschirmzeichens. Die Grundfarbeinstellung des COMMODORE 16 besteht aus dem weißen Hintergrund und der violetten Umrandung sowie dem in Schwarz blinkenden CURSOR.

Die Farbe des CURSORs, und damit auch die Farbe jedes künftig geschriebenen Zeichens, kann mit Hilfe der Farbtasten geändert werden. An den Zifferntasten <1> bis <8> stehen abgekürzt je zwei Farben.

Gemeinsam mit der Taste <Control> und einer Zifferntaste werden die Farben der oberen Reihe erreicht. Mit der COMMODORE-Taste <C=> und dem gleichzeitigen Drücken einer Zifferntaste erreichen Sie die Farben der unteren Reihe. Mit den Farbtasten lassen sich ausschließlich die Farben der Bildschirmzeichen ändern.

Um die Farbe von Bildschirm-Hintergrund oder -Rand zu ändern, sind eigene BASIC-Befehle notwendig, über die zu einem späteren Zeitpunkt ausführlich berichtet wird.

Nachstehend eine Zusammenfassung in Tabellenform:

	GEDRÜCKTE TASTE	plus +	TASTE	ergibt: =	
1.	<Control>	+	<1/Blk>	=	CURSOR/ZEICHEN in SCHWARZ
2.	<Control>	+	<2/Wht>	=	CURSOR/ZEICHEN in WEISS
3.	<Control>	+	<3/Red>	=	CURSOR/ZEICHEN in ROT
4.	<Control>	+	<4/Cyn>	=	CURSOR/ZEICHEN in ZYAN
5.	<Control>	+	<5/Pur>	=	CURSOR/ZEICHEN in PURPUR
6.	<Control>	+	<6/Grn>	=	CURSOR/ZEICHEN in GRÜN
7.	<Control>	+	<7/Blu>	=	CURSOR/ZEICHEN in BLAU
8.	<Control>	+	<8/Yel>	=	CURSOR/ZEICHEN in GELB
1.	<C=>	+	<1/Orng>	=	CURSOR/ZEICHEN in ORANGE
2.	<C=>	+	<2/Brn>	=	CURSOR/ZEICHEN in BRAUN
3.	<C=>	+	<3/Yl Grn>	=	CURSOR/ZEICHEN in GLB/GRN
4.	<C=>	+	<4/Pink>	=	CURSOR/ZEICHEN in ROSA
5.	<C=>	+	<5/Bl Grn>	=	CURSOR/ZEICHEN in BLA/GRN
6.	<C=>	+	<6/L Blu>	=	CURSOR/ZEICHEN in HL/BLAU
7.	<C=>	+	<7/D Blu>	=	CURSOR/ZEICHEN in DK/BLAU
8.	<C=>	+	<8/L Grn>	=	CURSOR/ZEICHEN in HL/GRN

2.4 GRAFIK-TASTEN

An jeder Buchstabentaste (und einigen anderen Tasten) des COMMODORE 16 sind nebeneinander zwei Grafik-Symbole aufgemalt. Diese Tasten stellen daher auch die Grafiktasten dar.

Nach dem Einschalten befindet sich der COMMODORE 16 im Großbuchstabenmodus, d.h. jede der oben beschriebenen Tasten schreibt, nachdem sie angetippt wird, einen Großbuchstaben bzw. das auf der Taste aufgemalte Zeichen auf den Bildschirm. Wird jedoch die Taste <Shift> oder die COMMODORE Taste <C=> gleichzeitig mit einer dieser Tasten gedrückt, so erscheint das zugehörige Grafikzeichen auf dem Bildschirm.

* Die Taste <Shift> bringt, gleichzeitig mit einer Buchstabentaste gedrückt, das entsprechende r e c h t e Grafikzeichen auf den Bildschirm.

* Die COMMODORE-Taste <C=> bringt, gleichzeitig mit einer Buchstabentaste gedrückt, das entsprechende l i n k e Grafikzeichen auf den Bildschirm.

Insgesamt lassen sich so über 60 Grafikzeichen, wie über den entsprechenden Tasten aufgemalt, auf dem Bildschirm darstellen.

	GEDRÜCKTE TASTE	plus +	TASTE	ergibt: =	
1.	<Shift>	+	<Buchst>	=	R e c h t e s Grafikzeichen
2.	<C=>	+	<Buchst>	=	L i n k e s Grafikzeichen

Durch Aneinanderreihen dieser Grafiksymbole, ähnlich dem Bauen mit Spielbausteinen, lassen sich die unterschiedlichsten Bilder, Tabellen und Zeichnungen entwerfen. Zur besseren Übersicht können die einzelnen Felder noch verschieden eingefärbt werden. Auch hier macht Übung den Meister! Im Teil 5 steht noch mehr über die Grafiksymbole.

Wenn Sie sich im Schreibmaschinenmodus befinden, so sind ausschließlich die l i n k e n Grafiksymbole (mit der COMMODORE-Taste <C=> und der entsprechenden Buchstabentaste) erreichbar. Diese links aufgemalten Grafikzeichen eignen sich besonders für Tabellen, grafische und kaufmännische Darstellungen.

```
***************************
*  2.5 FUNKTIONS-TASTEN   *
***************************
```

Im linken oberen Teil der Tastatur des COMMODORE 16 befinden sich vier (längliche) Funktionstasten, mit deren Benutzung sich viel Zeit sparen läßt. Durch einfaches Antippen dieser Tasten entstehen ganze Befehlszeilen auf dem Bildschirm. Diese Funktionstasten sind mit <F1/F4>, <F2/F5>, <F3/F6> und <Help/F7> beschriftet.

Die Funktionen <F1/>, <F2/>, <F3/> und <Help/> werden durch Drücken der entsprechenden Taste aufgerufen.

Die Funktionen </F4>, </F5>, </F6> und </F7> werden durch gleichzeitiges Drücken der Taste <Shift> und der entsprechenden Taste aufgerufen.

Und hier die Zusammenstellung aller möglicher Funktionen:

F1 Lädt einen der Grafik-Modi, wenn Sie die entsprechende Nummer hinzufügen und die Taste <RETURN> drücken.
Z.B. GRAPHIC 2 = geteilter Bildschirm mit einem Bildschirmmuster mit hoher Auflösung.

F2 Schreibt den Befehl DLOAD" auf dem Bildschirm und Sie tippen nur noch den Programmnamen dazu. Nach Drücken der Taste <Return> wird das Programm - sofern es sich auf der Diskette befindet - in den COMMODORE 16 geladen.

F3 Listet das Inhaltsverzeichnis der auf Diskette abgespeicherten Files am Bildschirm.

F4 Löscht den Bildschirm (auch im Grafik-Modus).

F5 Schreibt den Befehl DSAVE" auf den Bildschirm und Sie tippen nur noch den Programmnamen dazu. Nach Drücken der Taste <Return> wird das Programm - sofern es sich im Speicher des COMMODORE 16 befindet - auf Diskette abgespeichert.

F6 Bewirkt den Start eines im Speicher befindlichen Programms.

F7 Listet ein im Speicher befindliches Programm am Bildschirm.

Help läßt, nach dem Auftreten eines Fehlers, die fehlerhafte Programmzeile blinken.

Jede der Funktionstasten kann nach eigenem Ermessen, so oft dies gewünscht wird, neu definiert werden. Zur Neubelegung dient der Befehl 'KEY' (Detaillierte Erklärung erfolgt bei den BASIC-Befehlen im BASIC 3.5 Lexikon).

Um die momentane Belegung aller acht Funktionstasten auf einen Blick zu sehen, geben Sie den BASIC-Befehl 'KEY' ein und drücken die Taste <Return>. Auf dem Bildschirm wird daraufhin eine Liste aller acht Funktionstastenbefehle aufgelistet. Die Standard-Belegung des COMMODORE 16 ist wie folgt:

KEY und Taste <Return>

```
KEY 1,"GRAPHIC"
KEY 2,"DLOAD"+CHR$(34)
KEY 3,"DIRECTORY"+CHR$(13)
KEY 4,"SCNCLR"+CHR$(13)
KEY 5,"DSAVE"+CHR$(34)
KEY 6,"RUN"+CHR$(13)
KEY 7,"LIST"+CHR$(13)
KEY 8,"HELP"+CHR$(13)

READY.
```

Alle Funktionstasten können sowohl von BASIC-Programmen aus als auch im Direktmode neu belegt werden und behalten diesen Status bis zur erneuten Abänderung bzw. bis zum Ausschalten des COMMODORE 16 bei. Mit jedem Neueinschalten des Computers werden die Funktionstasten nach dem oben gelisteten Schema automatisch belegt.

DIE 'HELP'-TASTE

Programmfehler werden vom Betriebssystem des COMMODORE 16 erst während des Programmlaufs (Start mit Befehl 'RUN') erkannt und gemeldet. Die Erklärung jeder dieser möglichen Fehlermeldungen finden Sie im ANHANG A dieses Bedienungshandbuchs. Sofortige Hilfe bekommen Sie jedoch, wenn Sie die Funktionstaste <Help> drücken:

Die Programmzeile mit dem Fehler wird angezeigt, und zwar in blinkender Darstellung. Ein Beispiel einer Bildschirmmeldung:

?SYNTAX ERROR IN 10 READY.	Meldet der COMMODORE 16. Sie drücken daraufhin die Taste <Help/F7>.
HELP	
10 PRONT "COMMODORE COMPUTERS" READY.	Die Zeile mit dem falsch geschriebenen 'PRINT'-Befehl erscheint blinkend auf dem Bildschirm.

```
********************************
*                              *
*  T  E  I  L    3             *
*                              *
********************************
*                              *
*  B  E  N  U  T  Z  U  N  G   *
*                              *
*  D  E  R                     *
*                              *
*  S  O  F  T  W  A  R  E      *
*                              *
********************************
```

* EINLEITUNG
  ****
* STECKMODULE (CARTRIDGES)
  ****
* KASSETTEN
  ****
* DISKETTEN
  ****

****
* 3.1 EINLEITUNG *
****

Jeder Computer benötigt seine 'Software', also jene Programme, die eingegeben und gestartet, erst die Freude, Ablenkung und Unterstützung eines Computers bieten. Die 'Hardware' - in unserem Fall der COMMODORE 16 - kann Software in vielerlei Form verarbeiten:

aus dem STECK-MODUL = 'CARTRIDGE',
von der KASSETTE (bespielt),
von der DISKETTE (bespielt).

Die zur Auswahl stehende Palette an COMMODORE 16 - Software wächst ständig, wobei Ihr COMMODORE-Händler Sie nicht nur über die neuesten Produkte auf dem Laufenden hält. Vor allem wird er Sie auch mit den Möglichkeiten und Eigenschaften vorhandener Softwarepakete im einzelnen vertraut macht.

Neben der fertig angebotenen Software in Modul-, Kassetten- oder Disketten-Form können selbstverständlich auch 'Eigenproduktionen' auf Kassette oder Diskette abgespeichert und damit gesichert werden.

****
* 3.2 STECK-MODULE (CARTRIDGES) *
****

LADEN VON STECK-MODUL-PROGRAMMEN
****

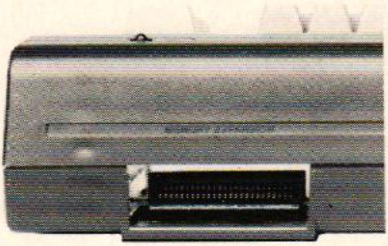
COMMODORE bietet ein umfangreiches Sortiment an MODUL-Software für den COMMODORE 16. Der Bereich erstreckt sich von Privat-Software über Lernprogramme bis zur Geschäfts-Software, ganz abgesehen von der großen Auswahl an Spielprogrammen.

Für den Einsatz der MODUL-Software sind keine zusätzlichen Peripheriegeräte notwendig. Im einzelnen ist folgendermaßen vorzugehen:

STUFE 1 Der Computer muß vor Einsatz des Moduls unbedingt ausgeschaltet sein!

WICHTIG: Modulwechsel bei eingeschaltetem Computer kann das Modul und den COMMODORE 16 beschädigen!

STUFE 2 Das Steckmodul wird mit der Beschriftung nach oben in den mit 'MEMORY EXPANSION' bezeichneten Schlitz an der Rückseite des COMMODORE 16 - ohne Gewaltanwendung - stecken.



MEMORY-EXPANSION-SCHLITZ FÜR STECKMODULE

STUFE 3 Erst jetzt dürfen Sie den COMMODORE 16 einschalten.

STUFE 4 Beginnen Sie mit dem Programm gemäß Anleitung - entweder mit einem Start-Befehl oder einem bestimmten Tastendruck.

****
* 3.3 KASSETTEN *
****

LADEN VON KASSETTEN-PROGRAMMEN
****

DATASSETTE MIT PROGRAMM-KASSETTEN

Eine Auswahl an Software ist für den COMMODORE 16 auch auf Kassetten lieferbar. Diese Kassetten sehen wie handelsübliche Musik-Kassetten aus, nur sind statt der Audiosignale Digitalsignale abgespeichert.

Kassetten-Software funktioniert in Verbindung mit dem COMMODORE 16 wie die Steckmodul-Software - jedoch ist ein Zusatzgerät (ein sog. Peripheriegerät) notwendig: die 'DATASSETTE'.

Die DATASSETTE ist ein Digital-Kassettenrekorder, mit dem Programme von Kassette in den COMMODORE 16 geladen bzw. auch Programme aus dem Computer auf handelsüblicher Musikkassette gespeichert werden können. Wie beispielsweise eigene Programme abgespeichert werden können, erfahren Sie im nächsten Abschnitt.

Der schrittweise Ablauf beim Laden von Kassetten-Software ist für fertig gekaufte und für selbsterstellte Software bzw. Programme der gleiche.

STUFE 1 Kassette mit der Beschriftung nach oben in das Kassettenfach einlegen und das Fach schließen.

STUFE 2 Befindet sich das Band nicht am Anfang, so drücken Sie auf der DATASSETTE die Taste <REW> (='REWIND ... Rückspulen) und spulen bis zum Anschlag (des Bandes) zurück - dann die Rekorder-Taste <STOP> drücken.

ANMERKUNG: Stellen Sie bei dieser Gelegenheit das Zählwerk der DATASSETTE auf '000'.

STUFE 3 Schreiben Sie in eine neue Bildschirmzeile den Befehl:

LOAD und drücken danach die Taste <Return>
Der COMMODORE 16 meldet auf dem Bildschirm:

PRESS PLAY ON TAPE worauf Sie ...

STUFE 4 ... die Rekorder-Taste <PLAY> drücken; der Bildschirm wird 'leer' und die DATASSETTE läuft an.

Findet der Rekorder ein abgespeichertes Programm auf der durchlaufenden Kassette, so kommt die Meldung (am Schirm):

FOUND ((Name des Programms))

STUFE 5 Soll besagtes Programm in den COMMODORE 16 geladen werden, so genügt ein Druck auf die COMMODORE-Taste <C=>, und der Ladevorgang beginnt (Bildschirm wird 'leer', DATASSETTE läuft erneut an). Wenn man nichts tut, passiert nach einer kurzen Pause automatisch dasselbe.

ANMERKUNG: Mit jeder FOUND-Meldung empfiehlt es sich auch, den Zählwerk-Stand mit dem Programmnamen zu notieren.

Wird ein später abgespeichertes Programm gewünscht, so müssen Sie den Ladevorgang mit der <Run/Stop>-Taste abbrechen und neu starten.

STUFE 6 Wurde ein Programm in den COMMODORE 16 geladen, dann erscheint der Bildschirminhalt wieder mit der Meldung:

READY.
■ CURSOR (blinkend)

Mit dem in eine neue Bildschirmzeile eingetippten Befehl

RUN und Druck auf die Taste <Return> starten Sie das geladene Programm.

Den Lade- bzw. Suchvorgang können Sie jederzeit mit Druck auf die Taste <Run/Stop> des COMMODORE 16 und nachfolgendem, einmaligen Drücken der Taste <STOP/EJECT> der DATASSETTE unterbrechen.

Geladene Basic-Programme können gelistet und geändert werden.

LADEN VON BESTIMMTEN KASSETTEN-PROGRAMMEN

Um ein bestimmtes Programm in den COMMODORE 16 zu laden, spulen Sie - sofern genaue Aufzeichnungen dafür vorhanden - zuerst die Kassette bis zum gewünschten Zählwerkstand vor bzw. zurück. Der Befehlsablauf ist bis zur STUFE 2 identisch mit der Beschreibung im vorangegangenen Abschnitt.

Wenn Sie z.B. das Programm mit dem Namen LEKTION laden wollen:

STUFE 3 Schreiben Sie in eine neue Bildschirmzeile den Befehl:

LOAD "LEKTION"
und drücken danach die Taste <Return>.
Der COMMODORE 16 meldet auf dem Bildschirm:

PRESS PLAY ON TAPE worauf Sie ...

STUFE 4 ... die Rekorder-Taste <PLAY> drücken; der Bildschirm wird 'leer' und die DATASSETTE läuft an.

Findet der Rekorder das abgespeicherte Programm auf der durchlaufenden Kassette, so kommt die Meldung (am Schirm):

FOUND LEKTION

STUFE 5 Ein Druck auf die COMMODORE-Taste <C=>, und der Ladevorgang beginnt (Bildschirm wird 'leer', DATASSETTE läuft erneut an).

STUFE 6 Es erscheint der Bildschirminhalt wieder mit der Meldung:

READY.

■ CURSOR (blinkend)

Mit dem in eine neue Bildschirmzeile eingegebenen Befehl

RUN und Druck auf die Taste <Return> startet das geladene Programm.

SPEICHERN VON PROGRAMMEN AUF KASSETTE
****

Eigene oder selbst eingegebene Programme können (und müssen) für spätere Modifikation oder erneuten Einsatz abgespeichert ('SAVE') werden. Um auf Kassette abzuspeichern, gehen Sie in folgenden Stufen vor:

STUFE 1 Suchen Sie eine freie Bandstelle und notieren Sie Zählwerkstand und Programmnamen. Das Suchen kann vom Verify-Befehl unterstützt werden (siehe BASIC 3.5 LEXIKON).

STUFE 2 Schreiben Sie in eine freie Bildschirmzeile den Befehl:

SAVE "Programmname" wobei der Programmname beliebigen Inhalts, jedoch max. 16 Zeichen lang sein darf.

Danach die Taste <Return> drücken.

Es kommt die Bildschirm-Meldung:

PRESS RECORD AND PLAY ON TAPE worauf Sie ...

STUFE 3 ... die Rekorder-Taste <REC> (Aufnahme) niederdrücken, wobei die danebenliegende Taste <PLAY> ebenfalls einrastet. Der Bildschirm wird 'leer' und die DATASSETTE läuft an.

STUFE 4 Wurde ein Programm aus dem COMMODORE 16 abgespeichert, dann erscheint der Bildschirminhalt wieder mit der Meldung:

READY.

■ CURSOR (blinkend)

BEISPIELE von SAVE-Befehlen für Kassettenbetrieb:

SAVE "HEUTE"	womit das abgespeicherte Programm 'HEUTE' heißt.
SAVE "MUSIK"	womit das abgespeicherte Programm 'MUSIK' heißt.

ANMERKUNG: Achten Sie stets darauf, ab welcher Bandstelle Sie abspeichern. Bei manchen Kassetten kann durch ein zu langes 'Vorspannband' (nicht magnetisch) der Anfang des abzuspeichernden Programms verloren gehen!

```
********************
*  3.4  DISKETTEN  *
********************
```

DISKETTENLAUFWERK UND DISKETTE

Disketten sind als Datenträger nicht nur einfach in der Anwendung sondern auch wesentlich schneller im Datentransfer. Diskettenlaufwerk und Diskette sind jedoch mit großer Sorgfalt zu behandeln. Wenn von Disketten-Laufwerk, 'Floppy Disk Drive' oder kurz 'Floppy' gesprochen wird, ist stets ein und dasselbe System gemeint.

Im Gegensatz zur DATASSETTE ist bei der FLOPPY nur die Diskette einzuschieben und das Laufwerk zu schließen; weitere Tasten sind am Laufwerk nicht vorhanden; es folgt nur noch die Befehlseingabe: LOAD oder SAVE bzw. DLOAD oder DSAVE.

An der Frontseite der Diskettenstation befinden sich zwei kleine Signallämpchen (LEDs). Das grüne Lämpchen signalisiert den Einschaltzustand (EIN = grün), und mit dem roten Lämpchen werden zwei Informationen mitgeteilt:

a) Es leuchtet, solange das Laufwerk für einen Lade- oder Speichervorgang läuft.

b) Es beginnt zu blinken (nachdem das Laufwerk stehen blieb), wenn mit Diskette oder Laufwerk ein Problem auftritt.

LADEN VON DISKETTEN-PROGRAMMEN
****

STUFE 1 Stellen Sie sicher, daß die Diskettenstation angeschlossen und eingeschaltet ist.

STUFE 2 Einsetzen der Diskette:
Dabei ist darauf zu achten, daß das Disketten-Etikett nach oben zeigt und zuletzt im Laufwerkschlitz verschwindet (Die Einkerbung für den Schreibschutz befindet sich an der linken Diskettenseite. Bei vor Beschreiben geschützter Diskette ist diese Einkerbung zugeklebt!).
Die Diskette muß vollständig, bis zum Anschlag, eingeschoben werden.

STUFE 3 Mit dem 'Tür'-Verschluß wird die Diskette im Laufwerk verriegelt.

STUFE 4 Schreiben Sie in eine neue Bildschirmzeile den Befehl:

DLOAD "Programmname" (Name des zu ladenden Programms)

Um Zeit zu sparen, kann auch FUNKTIONSTASTE </F2> gedrückt, dann der Programmname und Anführungszeichen (1x) geschrieben werden.

STUFE 5 Taste <Return> drücken ...

Die Floppy-Disk startet, und der COMMODORE 16 meldet auf dem Bildschirm:

SEARCHING FOR ((Programmname))

FOUND ((Programmname))

LOADING

READY.

■ CURSOR (blinkend)

STUFE 6 Mit dem in eine neue Bildschirmzeile eingetippten Befehl:

RUN und Druck auf die Taste <Return> startet das geladene Programm.

Sollte nach dem Ladevorgang (DLOAD) am Diskettenlaufwerk die rote Lampe blinken, dann trat ein Fehler auf, der mit folgendem Befehl abgefragt werden kann:

?DS$ und Taste <Return>.
Die auf dem Bildschirm erscheinende Fehlermeldung können Sie anhand der DOS-Fehlermeldungen (siehe Anhang) interpretieren.

Weitere Beispiele für den DLOAD-Befehl:

DLOAD "*"	Es wird das erste Programm des Disketten-Inhaltsverzeichnisses geladen.
DLOAD "FILES"	Es wird das Diskettenprogramm mit dem Filenamen 'FILES' geladen.
DLOAD "SOF*"	Es wird das erste Programm im Disketten-Inhaltsverzeichnis ('Directory') geladen, dessen Filename mit 'SOF' anfängt.
DLOAD "?AME"	Es kann z.B. sowohl das Programm 'NAME' oder auch 'DAME' geladen werden. Das '?' steht für ein beliebiges Zeichen. Das erste dazu passende Programm wird geladen. Ausschlaggebend ist die Reihenfolge der Progamme im Directory der Diskette.

FORMATIEREN ('HEADERN') EINER DISKETTE
****

Jede neue Diskette muß vor der Erst-Inbetriebnahme ein bestimmtes Format aufge'drückt' bekommen. Dieser auch als 'HEADERN' (= die Diskette mit einem Kopf, einem Titel versehen) bezeichnete Vorgang kann auch bei bereits gebrauchten Disketten angewendet werden - um sie zu löschen.

WICHTIG: Der FORMATIER- oder HEADER-Vorgang schreibt über die gesamte Diskette ein neues 'Muster'; bei bereits beschriebenen Disketten geht dadurch der gesamte Disketteninhalt v e r l o r e n!

Um eine Diskette zu formatieren, geben Sie folgenden Befehl:

HEADER "Disketten-Name",Ug,Iid,Dl

l = Laufwerk-Nummer (0/1) (Auch bei Einzel-Laufwerken!)

id = ID: Zweistelliges Kürzel, Buchstaben u./o. Ziffern; sollte von Diskette zu Diskette verschieden sein!

g = Geräteadresse: Normalerweise bei der Floppy die Zahl 8 (Optional, kann weggelassen werden).

Mit dem Disketten-Namen benennen Sie die Diskette; er darf bis zu 16 Zeichen (alpha/numerisch) lang sein.

Sobald die Taste <Return> gedrückt wird, erscheint am Bildschirm die Sicherheits-Abfrage:

ARE YOU SURE?

Diese Sicherheitsabfrage gibt dem Benutzer eine zusätzliche Kontrolle. Eine nochmalige Überprüfung der zu formatierenden Diskette ist daher ratsam, denn mit dem 'HEADER'-Vorgang sind eventuelle Daten auf der Diskette unrettbar verloren. Eine Maßnahme, die nicht für neue, noch unbeschriebene Disketten gilt, sondern bei bereits beschriebenen Disketten die letzte Rettung bedeuten kann.

Erst wenn Sie sicher sind, daß diese Diskette formatiert werden soll, geben sie 'Y' oder 'YES' ein und drücken erneut die Taste <Return>. Mit jeder anderen Eingabe brechen Sie den 'HEADER'-Vorgang ab.

Noch ein paar Beispiele für HEADER-Befehle:

```
HEADER "BRIEFE",U8,I07,D0          oder
HEADER "BUCHHALTUNG",IS3,D0      , dagegen:
HEADER "TESTDISK",D0              (ohne ID !)
       löscht nur den Inhalt der Diskette,
       ohne zu formatieren.
```

SPEICHERN VON PROGRAMMEN AUF DISKETTE

Ein neu geschriebenes Programm sollte noch vor dem ersten Testlauf stets abgespeichert werden, da es sowohl beim Ausschalten des COMMODORE 16 als auch beim Laden eines anderen Programms verloren geht. Geänderte Programme können über die alte Version oder als neue Version - und mit neuem Namen - abgespeichert werden.

So wird ein neues Programm auf Diskette abgespeichert:

DSAVE "Programm-Name" ... in eine neue Bildschirmzeile schreiben und Taste <Return> drücken.

Um Zeit zu sparen, kann auch FUNKTIONSTASTE </F5> gedrückt, dann der Programmname und einmal Anführungszeichen geschrieben werden. Anschließend die Taste <Return> drücken.

Während des bzw. nach dem Speichervorgang kommen folgende Bildschirmmeldungen:

SAVING Programm-Name

OK

READY.

■ CURSOR (blinkend)

Ein weiteres Beispiel:

DSAVE "MEIN PROGRAMM" Programm-Name kann max. 16 Zeichen lang sein.

Sollte nach dem Speichervorgang (DSAVE) am Diskettenlaufwerk die rote Lampe blinken, dann trat ein Fehler auf, der mit folgendem Befehl abgefragt werden kann:

?DS$	und Taste <Return>.
	Die auf dem Bildschirm erscheinende Fehlermeldung kann anhand der DOS-Fehlermeldungen (siehe Anhang) interpretiert werden.

INHALTSVERZEICHNIS ('DIRECTORY') DER DISKETTE

Mit jedem SAVE-Vorgang auf Diskette erfolgt automatisch der Eintrag des Programmnamens (FILEname) in das Inhaltsverzeichnis (DIRECTORY) der Diskette. Dieses Inhaltsverzeichnis kann mit dem DIRECTORY-Befehl am Bildschirm gelistet werden:

DIRECTORY	und Taste <Return> drücken.
	Um Zeit zu sparen, kann auch FUNKTIONSTASTE <F3> gedrückt werden.

Mit dem RETURN-Befehl erscheint das gesamte Inhaltsverzeichnis der Diskette auf dem Bildschirm. Einige Befehls-Beispiele:

DIRECTORY "MEIN*"	Listet sämtliche Filenamen aus dem Inhaltsverzeichnis der Diskette, die mit 'MEIN' beginnen.
DIRECTORY "*=PRG"	Listet sämtliche PROGRAMM-Files der Diskette.
DIRECTORY "*=SEQ"	Listet sämtliche SEQUENTIELLEN Daten-Files der Diskette.

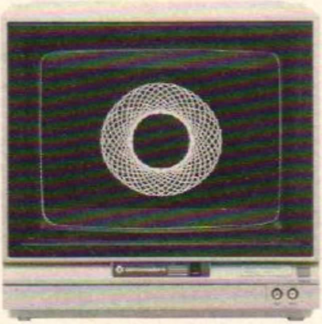
```
*******************************
*                             *
*  T  E  I  L     4           *
*                             *
*******************************
*                             *
*                             *
*                             *
*  E  R  S  T  E              *
*                             *
*  S  C  H  R  I  T  T  E     *
*                             *
*******************************
```


* EINLEITUNG
  ****
* BILDSCHIRM
  ****
* REVERSE DARSTELLUNG
  UND FARBÄNDERUNG
  ****
* ERSTE PROGRAMMIERSCHRITTE
  ****
* BEFEHLS-EINGABE
  ****
* FEHLERBEHANDLUNG
  ****
* BILDSCHIRM- UND ARBEITS-
  SPEICHER LÖSCHEN
  ****
* BILDSCHIRM-FENSTER
  ****

****
* 4.1 EINLEITUNG *
****

Ziel dieses Handbuchteils ist es, Sie mit einigen Eigenschaften und Fähigkeiten des COMMODORE 16 vertraut zu machen. Des weiteren werden erste Programmierhilfen für Ihren Computer gegeben.

****
* 4.2 BILDSCHIRM *
****



Der Bildschirm stellt eine der Schnittstellen des COMMODORE 16 zur Außenwelt dar. Jede Eingabe über die Tastatur, jede Meldung des Betriebssystems, jeder Ladevorgang usw. wird am Bildschirm angezeigt. Dabei besteht kein Unterschied zwischen einem S/W-Monitor (Schwarz/Weiß-Bildschirm) und einem Farbfernseher.

Der COMMODORE 16 beschreibt pro Zeile max. 40 Spalten, d.h. von der linken bis zur rechten Bildschirmkante können bis zu 40 Zeichen geschrieben werden, dann 'springt' der CURSOR in die nächste Zeile. Insgesamt können auf diese Weise 25 Zeilen beschrieben werden.

Der vollständig beschriebene Bildschirm umfaßt daher 1000 Zeichen (40 Zeichen x 25 Zeilen).

Der Bildschirmaufbau des COMMODORE 16 setzt sich aus drei Elementen zusammen:

+ den Buchstaben, Ziffern, Sonderzeichen und grafischen Zeichen,
+ der Hintergrund-Farbe,
+ der Rahmen-Farbe.

Mit dem Einschaltvorgang sind die Zeichen auf dem Bildschirm in Schwarz, die Hintergrund-Farbe ist Weiß, und die Rahmen-Farbe erscheint in hellem Violett. Daß die einzelnen Farben unter Einsatz der Taste <Control> sowie der COMMODORE-Taste <C=> geändert werden können, wurde bereits im Kapitel 2.3 FARB-TASTEN beschrieben. Insgesamt stehen 16 Farben für die Zeichen-Einfärbung zur Auswahl. Und es gibt weitere Möglichkeiten, die Zeichendarstellung interessanter zu gestalten:

****
* 4.3 REVERSE DARSTELLUNG UND FARBÄNDERUNG *
****

Neben der Farbänderung der CURSOR-Farbe und damit der Zeichenfarbe auf dem Bildschirm können Zeichen- und Hintergrundfarbe auch umgekehrt (REVERS) dargestellt werden. Dies bedeutet, daß schwarze Zeichen auf weißem Hintergrund mit dem REVERS-Befehl zu weißen Zeichen auf schwarzem Hintergrund werden (Taste <Control> und Taste <Rvs/On> gleichzeitig drücken). Experimentieren Sie mit diversen Farbkombinationen, indem Sie folgendermaßen vorgehen:

STUFE 1 Taste <Control> und Taste <Rvs/On> gleichzeitig drücken.

STUFE 2 Als nächstes halten Sie die LEER-Taste gedrückt.

STUFE 3 Solange Sie nun die LEER-Taste gedrückt halten, entsteht eine Farbzeile in der Farbe des zuletzt geschriebenen Zeichens. Ist eine Zeile voll, so wird das Farbband in der nächsten Zeile fortgesetzt.

STUFE 4 Die LEER-Taste loslassen - aber nicht die Taste <Return> drücken!

STUFE 5 Wenn Sie als nächstes die Taste <Control> und gleichzeitig eine der noch nicht auf dem Schirm verwendeten Farben drücken, dann wechselt der CURSOR in die gedrückte FARB-tastenfarbe um.

STUFE 6 Drücken Sie nun die LEER-Taste erneut, so wird eine weitere Farbzeile in der neuen Farbe entstehen. Setzen Sie den Farbwechsel durch Betätigen der Taste <Control> und der COMMODORE-Taste <C=> fort. Auf diese Weise (LEER-Taste drücken) werden neue Farbzeilen entstehen.

STUFE 7 Mit der Tastenkombination <Control> und <Rvs/Off> (gleichzeitig gedrückt) heben Sie den REVERS-Modus wieder auf. Gleiches passiert mit Druck auf die Taste <Return>

ZUSAMMENFASSUNG:
****

Buchstaben, Ziffern und Zeichen können auch in Negativschrift (REVERS) dargestellt werden. Diese Schreibart eignet sich z.B. gut für Überschriften. Auch einzelne Worte, Zahlen oder Passagen lassen sich auf diese Art besonders hervorheben. Graphikzeichen sind mit dieser Methode ebenfalls REVERS darstellbar.

Am Beispiel eines 'Rennstreifen'-Musters in Farbe läßt sich das Vorangegangene demonstrieren:

Sie wählen eine Farbe aus und schreiben anschließend eine Zeile mit dem Zeichen <¯> (Taste <Shift> und Taste <E>) voll.

Dann schalten Sie auf den REVERS-Modus um (Taste <Control> und Taste <Rvs/On> gleichzeitig drücken).

Füllen Sie eine weitere Zeile mit dem Zeichen <¯>, diesmal jedoch in REVERS-Darstellung.

Eine weitere Zeile wird nun mit dem Zeichen <_> (Taste <Shift> und Taste <R>) ebenfalls in REVERS-Darstellung gefüllt.

Jetzt wird der REVERS-Modus wieder abgeschaltet (Taste <Control> und Taste <Rvs/Off> gleichzeitig drücken).

Und nun wird eine letzte Zeile ebenfalls mit dem Zeichen <_> gefüllt - diesmal wieder im NORMAL-Modus.

Auf dem Bildschirm sehen Sie jetzt ein Rennstreifen-Muster in dem von Ihnen gewählten Muster und der gewählten Farbe.

```
************************************
*  4.4  ERSTE PROGRAMMIERSCHRITTE  *
************************************
```

Was ist eigentlich ein Programm? Wenn Sie einen erfahrenen Programmierer fragen, wird seine Antwort etwa so oder ähnlich lauten:

Ein Programm stellt eine Serie bestimmter Befehle dar, die den Computer in die Lage versetzen, ein gestecktes Ziel durch Abarbeiten der einzelnen Programmschritte zu erreichen.

Für einen Anfänger wird so etwas wahrscheinlich fürchterlich kompliziert klingen. Dabei stellen doch die nachfolgenden zwei Zeilen bereits ein vollständiges Programm dar! Tippen Sie diese beiden Zeilen exakt ein. Die Zahlen am Anfang der Zeilen müssen unbedingt mit eingegeben werden, da es sich um die sogenannten ZEILENNUMMERN handelt. Anhand der Zeilennummer wird dem Computer die Reihenfolge des Programmablaufs (in aufsteigender Zahlenfolge) vorgeschrieben. Zeilennummern dürfen Werte zwischen 0 und 63999 annehmen. Jede eingegebene Zeile muß durch Drücken der Taste <Return> abgeschlossen werden!

10 PRINT"ERSTER VERSUCH"		Mit dieser Zeile erhält der Computer den Befehl, die Worte ERSTER VERSUCH auf den Bildschirm zu schreiben.
20 GOTO 10		Der 'GOTO'-Befehl veranlaßt, in Verbindung mit einer Zeilenzahl, einen Programmsprung zu dieser Zeile.
RUN	...	und Taste <Return> startet obiges Programm.

Mit der Taste <Run/Stop> läßt sich das Programm anhalten.

Es stellt sich die Frage, warum der Computer ununterbrochen, bis zum Drücken der Taste <Run/Stop>, die Worte 'ERSTER VERSUCH' auf den Bildschirm schrieb?

Über den Befehl 'PRINT' wird dem Computer mitgeteilt, alles, was nach diesem Befehl zwischen Anführungszeichen (" ") geschrieben steht, auf dem Bildschirm auszudrucken. Erreicht - beim Abarbeiten des Programms - der Computer diese Befehlszeile, wird sie ausgeführt.

Beim Befehl 'GOTO 10' muß der Computer erneut die Zeile 10 an'springen' und den dort verzeichneten PRINT-Befehl erneut abarbeiten. Dieser geschlossene Vorgang wird SCHLEIFE (='LOOP') genannt. Weil die Schleife ohne weiteres Zutun ständig durchlaufen würde, spricht man von einer 'unendlichen Schleife'. Mit der Taste <Run/Stop> kann jedoch in diese (Endlos-) Schleife einge'brochen' werden und die Kontrolle des Computers wieder über die Tastatur ausgeübt werden.

Drücken Sie nach den folgenden Zeileneingaben stets abschließend die Taste <Return>:

NEW	... und Taste <Return> läßt den Computer alles in seinen Arbeits-Speicher eingegebene 'vergessen'. Nach dem NEW-Befehl ist Ihr COMMODORE 16 wieder frei für neue Aufgaben.
READY.	... wird nicht durch Sie eingegeben, sondern damit meldet sich der Computer bereit (=READY) für weitere Befehle oder Programme.

<Control> UND <Rvs/On> BZW. <Rvs/Off> PROGRAMMIERT

Werden die Tasten <Control> und <Rvs/On> bzw. <Rvs/Off> zwischen Anführungszeichen geschrieben, so erscheint anstelle der sofortigen Ausführung der Funktionen ein codiertes REVERS-Zeichen auf dem Bildschirm. Ein Beispiel:

```
10 PRINT"RCOMMODORE 16_"
```

Erste Stelle nach dem ersten Anführungszeichen = <Rev/On>.

Letzte Stelle vor dem zweiten Anführungszeichen = <Rev/Off>.

Wenn Sie 'RUN' eingeben und die Taste <Return> drücken, so wird der Schriftzug 'COMMODORE 16' in REVERS-Darstellung auf dem Bildschirm erscheinen, d.h. die in der Zeichenkette gespeicherten Funktionen Rev/On und Rev/Off werden beim Aufruf durch PRINT korrekt ausgeführt.

Dieses Prinzip der Darstellung zwischen Anführungszeichen gilt auch für andere Control-Funktionen, wie z.B. CURSOR-Bewegungen oder Farbänderungen. Durch das erste Anführungszeichen wird der Computer in den sog. Anführungszeichenmodus ('Quote-Modus') geschaltet. In dieser Betriebsart werden die von der Tastatur aufgerufenen Control-Funktionen durch reverse Platzhalter-Symbole dargestellt, die erst beim Aufruf durch PRINT die entsprechende Funktion auslösen. Durch das abschließende Anführungszeichen wird dieser Modus wieder ausgeschaltet.

Anmerkung: Sollten Sie einmal versehentlich in den Anführungszeichen-Modus geraten sein, so können Sie ihn wieder aufheben, indem Sie <Esc> mit anschließendem <O> eingeben.

<Control> UND <Flash/On> BZW. <Flash/Off> PROGRAMMIERT

Werden die Tasten <Control> und <Flash/On> bzw. <Flash/Off>

zwischen Anführungszeichen geschrieben, so erscheint ein entsprechendes REVERS-Zeichen auf dem Bildschirm. Ein Beispiel:

10 PRINT"▌|COMMODORE 16▀"	Erste Stelle nach dem ersten Anführungszeichen = <Flash/On>.
	Letzte Stelle vor dem zweiten Anführungszeichen = <Flash/Off>.

Wenn Sie 'RUN' eingeben und die Taste <Return> drücken, so wird der Schriftzug 'COMMODORE 16' auf dem Bildschirm erscheinen und ständig blinken.

```
*************************
* 4.5 BEFEHLS-EINGABE *
*************************
```

Wie Sie sicher bemerkt haben, werden einmal Befehle im DIREKT-Modus (z.B. der Befehl 'RUN' oder 'NEW') eingegeben. Zum anderen werden Befehle in numerierte Programmzeilen geschrieben, im sog. INDIREKT- oder PROGRAMM-Modus.

In beiden Modi basiert die Kommunikation mit dem Computer auf Schlüsselwörtern (vereinbarten Termini) der Programmiersprache BASIC. Diese Computersprache BASIC ist - einschließlich aller Termini - in der Version 3.5 im COMMODORE 16 fest installiert.

Im DIREKT-Modus reagiert der COMMODORE 16 auf jeden eingegebenen und per Taste <Return> abgesandten Befehl sofort mit dessen Ausführung. Die Alternative dazu ist der INDIREKT- oder PROGRAMM-Modus, bei dem die Befehlsfolge in numerierte Programmzeilen geschrieben werden muß. Diese Folge von Programmzeilen wird mit dem DIREKT-Befehl 'RUN' (bei der niedrigsten Zeilennummer beginnend) gestartet. Computerprogramme laufen grundsätzlich im PROGRAMM-Modus. Mehrere Befehle dürfen, durch Doppelpunkte getrennt, in einer BASIC-Zeile stehen. Eine Programmzeile kann bis zu zwei Bildschirmzeilen (80 Zeichen) lang sein.

****
* 4.6 FEHLERBEHANDLUNG *
****

Auch beim Umgang mit dem Computer kommt es zu Fehlern. Damit ein Programm fehlerfrei abläuft, müssen eventuelle Fehler entdeckt und behoben werden. Eine Hilfe stellt dabei die Taste <Help> dar. Die häufigsten Fehler sind Schreibfehler bei der Eingabe; sie können sich auf das Programm sehr störend auswirken. Daher werden durch das Programmieren nicht nur Ihre Schreibmaschinenfähigkeiten merklich verbessert, sondern Sie werden auch viel Erfahrungen in der Fehlerkorrektur sammeln.

Es gibt mehrere Möglichkeiten, Tippfehler zu beheben. Dabei vergessen Sie nicht, daß jede Änderung - egal ob die Ausführung oder den Speicherbereich betreffend - dem Computer durch abschließende Betätigung der Taste <Return> mitgeteilt werden muß.

1. Korrektur (EDITieren) einer Zeile durch Überschreiben

   Bewegen Sie den CURSOR mit der CURSOR-Taste an die Stelle mit dem Fehler und überschreiben Sie ihn einfach.

   Ein Beispiel:

10 PRINT "ES IST ZEHN UHR MORGENS"	und Sie wollen auf 'ACHT' Uhr ändern, dann stellen Sie den CURSOR auf das 'Z' und
10 PRINT "ES IST ACHT UHR MORGENS"	überschreiben die 'ZEHN' einfach.

   ANMERKUNG: Dabei muß der CURSOR nicht an das Zeilenende geführt werden, um die Taste <Return> zu drücken. In diesem Fall kann, unabhängig von der CURSOR-Position, die Taste <Return> sofort nach der Fehlerkorrektur gedrückt werden, um die korrigierte Zeile an den Computer zu übergeben.

2. Korrektur (Editieren) einer Zeile durch Einfügen (INSerT)

Bewegen Sie den CURSOR mit der CURSOR-Taste an die Stelle, ab der Sie ein oder mehrere Zeichen einfügen wollen. Der CURSOR muß hinter der letzten korrekten Stelle stehen bleiben!

Halten Sie nun die Taste <Shift> gedrückt und betätigen Sie die Taste <Inst/Del> so oft, wie einzufügende Zeichen eingetragen werden sollen.

Ein Beispiel:

10 PRINT "CORE" — Es soll der fehlende Schriftzug 'OMMOD' eingefügt werden. Stellen Sie nun den CURSOR auf das 'O' ('O' blinkt REVERS). Danach die Taste <Shift> gedrückt halten und die Taste <Inst/Del> 5x betätigen.

Der Schriftzug-Rest 'ORE' wird 5x nach rechts verschoben, und in die entstandene Lücke können jetzt die fehlenden Buchstaben 'OMMOD' eingeschrieben werden.

10 PRINT "COMMODORE" ... lautet die korrigierte Zeile.

ANMERKUNG: Haben Sie zu wenig Plätze geöffnet, dann erweitern Sie nach oben beschriebener Methode.

Nach INSerT sind Sie in der erzeugten Lücke im Quote-Modus, CURSOR und andere Control-Zeichen werden also nicht ausgeführt!

Bei zu vielen Leerplätzen schließen Sie die verbleibende Lücke mit nachstehender Methode.

3. Korrektur (Editieren) einer Zeile durch Löschen (DELete)

Zeichen in einer Zeile werden gelöscht bzw. Lücken einer Zeile geschlossen, indem der CURSOR auf das erste Zeichen, das sich rechts von den zu löschenden bzw. zu schließenden Stellen befindet, positioniert (mit CURSOR- oder LEER-Taste) wird (Buchstabe unter dem CURSOR blinkt).

Die Taste <Inst/Del> wird anschließend so oft gedrückt, wie Stellen gelöscht werden sollen.

4. Korrektur (Editieren) einer Zeile durch Neueingabe

Jederzeit ist die Neueingabe einer oder mehrerer Programmzeilen möglich - auch nach dem 'RUN'-Befehl.

Mit der Neueingabe (bei gleicher Zeilennummer) wird die alte Programmzeile im Computer nach der Tasteneingabe <Return> von der neuen Zeile überschrieben. Die eventuell noch am Bildschirm stehende alte Zeile wird vom Computer ignoriert. Bei zwei Programmzeilen mit gleicher Zeilennummer übernimmt der Computer stets die zuletzt eingegebene Zeile in den Speicher.

Ein Beispiel:

```
10 COKOR 0,3
     !
     Fehler!
20 PRINT "COMMODORE 16"
```

Gehen Sie mit Tastendruck <Return> in eine neue Zeile und schreiben Sie die Programmzeile 10 einfach neu:

10 COLOR 0,3   und drücken Sie erneut die Taste <Return>.

Damit ist die erste Zeile 10 durch die zuletzt geschriebene Zeile 10 im Computer ersetzt. Mit dem 'LIST'-Befehl läßt sich dies sogleich - Zeile für Zeile direkt aus dem Speicher des Computers - auslesen. Vom Computer wurde jede Zeile entsprechend ihrer Zeilennummer eingeordnet. Ersetzte, alte Zeilen tauchen in diesem 'LISTing' nicht mehr auf.

Geben Sie also ein:

LIST   und Taste <Return> drücken ...

Auf den Bildschirm wird geschrieben:

```
10 COLOR 0,3

20 PRINT "COMMODORE 16"
```

Mit dem Befehl 'RUN' und Taste <Return> sehen Sie gleich die Auswirkungen dieses Programms. Mehr darüber im Teil 6 dieses Handbuchs, oder unter 'COLOR' im BASIC-Lexikon.

Ein Übungsbeispiel für den COMMODORE 16 stellt auch der generelle Austausch von Programmzeilen dar. Wird eine Programmzeile durch eine neue ersetzt, so kann diese neue Zeile ganz anders aufgebaut sein. Zum Beispiel kann anstatt der Korrektur der Zeile 10 (im letzten Beispiel) eine völlig neue Programmzeile eingegeben werden:

```
10 PRINT"COMMODORE 4 MAL COMMODORE 4  = "
```

Mit dem 'RUN'-Befehl sehen Sie, was das Programm nun bringt.

5. Korrektur (Editieren) durch Löschen (ERASE) einer Zeile

Um eine Programmzeile aus dem Programm und damit aus dem Speicher zu löschen, genügt es, die betreffende Zeilennummer dieser Programmzeile in eine neue Zeile zu schreiben und die Taste <Return> zu drücken. Obwohl besagte Zeile noch auf dem Bildschirm geschrieben stehen kann, ist sie im Computerprogramm gelöscht.

Mit dem 'LIST'-Befehl läßt sich dies sogleich - Zeile für Zeile direkt aus dem Speicher des Computers - auslesen. Vom Computer wurde jede Zeile entsprechend ihrer Zeilennummer eingeordnet. Ersetzte, alte Zeilen tauchen in diesem 'LISTing' nicht mehr auf:

Geben Sie also ein:

LIST und Taste <Return> drücken.

Auf den Bildschirm wird geschrieben:

```
10 PRINT "COMMODORE 4 MAL COMMODORE 4  = "

20 PRINT "COMMODORE 16"
```

Geben Sie nun ein:

10 und Taste <Return> drücken und erneut

LIST und Taste <Return> drücken.

Auf den Bildschirm wird geschrieben:

```
20 PRINT "COMMODORE 16"
```

```
*************************************************
*  4.7  BILDSCHIRM- UND ARBEITSSPEICHER LÖSCHEN  *
*************************************************
```

Im Verlauf Ihrer Programmierarbeit und Versuche wird sich der Bildschirm immer mehr füllen und Sie wünschen eine Be'reinigung' dieses Zustands. Nichts leichter, als wieder zu einem leeren Bildschirm zu kommen. Dazu gibt es mehrere Möglichkeiten. Einmal können Sie Bildschirm- und Arbeitsspeicher gleichzeitig löschen; Sie können aber auch nur den Bildschirm oder nur den Arbeitsspeicher löschen.

Eine, wenn auch die uneleganteste, Methode, den Bildschirm zu löschen, besteht darin, von der 'HOME'-Position aus die LEER-Taste (SPACE) zu drücken und den CURSOR über alle 1000 Felder laufen zu lassen. Aber es gibt bessere Methoden:

1. Drücken Sie gleichzeitig die Tasten <Shift> und <Clr/Home>. Der Bildschirm wird dadurch gelöscht und der CURSOR in die 'HOME'-Position (linke, obere Ecke des Bildschirms) positioniert. Der Arbeitsspeicher bleibt unverändert.

2. Im DIREKT-Modus den BASIC-Befehl 'SCNCLR' eintippen und die Taste <Return> drücken, worauf der Bildschirm gelöscht wird. Auch hierbei wird der Arbeitsspeicher nicht berührt.

3. Drücken Sie die Taste 'RESET' (an der rechten Seite des COMMODORE 16). Da bei dieser Methode nicht nur der Bildschirm, sondern auch der gesamte Speicher des COMMODORE 16 gelöscht wird, ist hier größte VORSICHT geboten!

4. Im DIREKT-Modus den BASIC-Befehl 'NEW' eintippen und die Taste <Return> drücken. Dieser Befehl bewirkt, daß der Programmspeicher des COMMODORE 16 wieder frei wird. Der Bildschirm wird dabei nicht gelöscht.

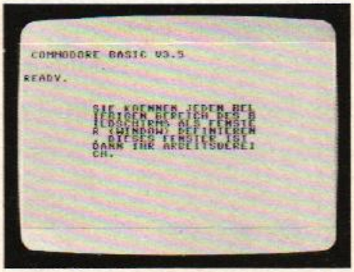
5. Die <RUN/STOP>-Taste gedrückt halten und gleichzeitig die 'Reset'-Taste auf der Seite betätigen. Der Rechner unterbricht auch laufende Maschinenprogramme und landet im Monitor TEDMON (siehe Anhang). Mit Eingabe von X und <Return> ist man ohne Programmverlust wieder im Normalzustand.

```
****************************
*  4.8  BILDSCHIRM-FENSTER  *
****************************
```

Es ist mit dem COMMODORE 16 möglich, Teile des Gesamtbildschirms als sogenannte Fenster ('WINDOWS') zu definieren. In einem Fenster kann dann von der Programmeingabe bis zum Programmlisting alles ablaufen, ohne den Bildschirminhalt außerhalb des Fensters zu beeinflussen. Dieses Fenster kann an jeder Stelle des Bildschirms eingerichtet werden.

Vier Stufen sind zur Einrichtung eines Fensters zu beschreiten:

1. Den CURSOR auf die linke, obere Fensterecke, die Sie definieren wollen, positionieren.
2. Die Taste <Esc> und danach die Taste <T> drücken.
3. Mit dem CURSOR die rechte, untere Fensterecke anwählen.
4. Die Taste <Esc> und dann die Taste <B> drücken.



BILDSCHIRM MIT EINGERICHTETEM FENSTER

Damit ist die Position des Fensters festgelegt. Alle Meldungen (PRINTs, Listings usw.) landen nun im Bildschirmfenster. Um das Fenster wieder aufzulösen, ist die Taste <Home> zweimal zu drücken. Der CURSOR steht danach in der linken, oberen Ecke ('HOME'-Position).

```
**********************************
*  4.9  ESCAPE-TASTENFUNKTIONEN  *
**********************************
```

Um sowohl das Fenster wie auch seine Inhalte manipulieren zu können, steht die Taste <Esc> zur Verfügung. Editier-Funktionen, wie Einsetzen und Löschen von Text, Durchlaufenlassen (SCROLLEN), und auch Veränderungen am Fenster selbst, werden durch Druck auf die Taste <Esc> und eine entsprechende zweite Taste (siehe nachstehende Tabelle) erreicht.

<ESC> & TASTE	FUNKTION
A	Automatisch einfügen
B	(Set Bottom) Fixiert an der gegenwärtigen CURSOR-position die rechte, untere Fensterecke
C	(Clear auto insert) Hebt automatisch einfügen auf
D	(Delete) Löscht eine Zeile an der CURSOR-Position
I	(Insert) Fügt eine Zeile an der CURSOR-Position ein
J	CURSOR wird an den Anfang der CURSOR-Positionszeile gesetzt
K	CURSOR wird an das Ende der CURSOR-Positionszeile gesetzt
L	Schaltet SCROLLING-Modus ein
M	Schaltet SCROLLING-Modus aus
N	Schaltet zur normalen Bildschirmgröße zurück und löscht den Bildschirm.
O	(Off) Hebt Einfüge-, Anführungszeichen-, Reverse- und Blink-Modus wieder auf
P	Löscht Bildschirmzeile vom Anfang bis zur CURSOR-Position.
Q	Löscht Bildschirmzeile ab der CURSOR-Position bis zum Ende.
R	Verkleinert das Bildschirm-Format und löscht den Bildschirm.
T	(Set Top) Fixiert an der gegenwärtigen CURSOR-Position die linke, obere Ecke des Fensters.
V	SCROLLEN des Bildschirminhalts nach oben
W	SCROLLEN des Bildschirminhalts nach unten
X	(Exit ESC) Befreit Sie aus dem ESCAPE-Modus nach versehentlicher Betätigung der <ESC>-Taste.

```
*******************************
*                             *
*  T E I L   5                *
*                             *
*******************************
*                             *
*  Z A H L E N                *
*                             *
*  U N D   R E C H E N -      *
*                             *
*  O P E R A T I O N E N      *
*                             *
*******************************
```

* EINLEITUNG
  ****
* ZAHLEN UND
  GRUNDRECHENFUNKTIONEN
  ****
* AUSFÜHRUNG VON
  RECHENOPERATIONEN
  ****
* WEITERE BEFEHLE ZUM
  BILDSCHIRMAUSDRUCK
  ****
* VARIABLE
  ****

```
********************
*  5.1 EINLEITUNG  *
********************
```

Um die mathematischen Fähigkeiten des COMMODORE 16 zu verstehen, muß man kein Mathematikprofessor sein. Zusätzlich zu den vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division ist der COMMODORE 16 auch für fortgeschrittene Rechenoperationen, wie Quadratwurzelziehen oder Sinuskurvenberechnungen, einzusetzen.

In diesem Teil des Handbuchs werden Sie die unterschiedlichsten Variablentypen und ihren richtigen Einsatz kennenlernen. Der COMMODORE 16 verarbeitet sowohl ganze Zahlen als auch Gleitkommazahlen mit bis zu 38 Stellen in Exponential-Schreibweise, wobei allerdings nur die ersten 9 Stellen berücksichtigt werden. Die Berechnungen können im DIREKT-Modus oder eingebunden in ein Programm erfolgen. Abschließend erhalten Sie eine kurze Anleitung, wie Sie selbstdefinierte Funktionen auf dem COMMODORE 16 implementieren können.

```
****************************************
*  5.2  ZAHLEN UND GRUNDRECHENFUNKTIONEN  *
****************************************
```

Der COMMODORE 16 ist auch wie ein Taschenrechner einsetzbar. Neben den Standard-Operatoren für Addition ('+') und Subtraktion ('-') verwendet der COMMODORE 16 - wie viele andere Computer - für Multiplikation das 'Sternchen' (Asterisk, '*') und für Division bzw. Bruchstrich den Schrägstrich ('/').

Die Grundrechenfunktionen stehen im DIREKT-Modus und im PROGRAMM-Modus zur Verfügung. Im PROGRAMM-Modus werden die Rechenoperationen z.B. in PRINT-Befehlen o h n e Anführungszeichen geschrieben.

ARITHMETISCHE OPERATOREN		LOGISCHE VERGLEICHSOPERATOREN	
Addition	+	Größer als	>
Subtraktion	-	Kleiner als	<
Division und Bruch	/	Gleich	=
Multiplikation	*	Größer oder gleich	>= oder =>
Exponent	↑	Kleiner oder gleich	<= oder =<
		Ungleich	<> oder ><

ANMERKUNG: Zahlen dürfen bei der Computereingabe k e i n Komma (',') enthalten - weder als (englische) Tausender-Trennung noch als Dezimalpunktersatz (21,00). Bei der Computereingabe bedeutet das Komma die Trennung zweier Zahlen; daher Eingabe von Zahlen so: 30359 oder so: 21.00

BRÜCHE UND DEZIMALSTELLEN

Ein Bruch kann so: .5

oder so dargestellt werden: 1/2

Im zweiten Fall wird der Computer das Ergebnis errechnen. Wenn die Rechenoperation in einen PRINT-Befehl eingebettet ist, wird das Ergebnis immer im Dezimalmodus bzw. als ganze Zahl ausgegeben.

Beispiel:

PRINT 139/493 + 5 und Taste <Return>:

5.28194726 ist das Ergebnis am Bildschirm.

DIE ' π '-TASTE
****

Ein Beispiel für eine Zahl, die sich nicht exakt als Bruch oder als Dezimalzahl mit endlich vielen Stellen schreiben läßt, ist die Zahl π (sprich:PI), die das Verhältnis des Kreisumfanges zu seinem Durchmesser ausdrückt. Die Zahl π ist mit einer Genauigkeit von acht Stellen nach dem Dezimalpunkt im COMMODORE 16 gespeichert und über die Taste < π > abrufbar. Ein Beispiel:

```
PRINT π /374            und Taste <Return>:
  8.39998036E-03        erscheint auf dem Bildschirm.
```

WISSENSCHAFTLICHE NOTATION
****

Was bedeutet eigentlich der Anhang 'E-03' in obigem Rechenbeispiel? Der COMMODORE 16 kann Zahlen im Bereich von -999 999 999 bis +999 999 999 direkt verarbeiten. Über neun Stellen hinausgehende Zahlen werden automatisch auf die ersten neun signifikanten Stellen (d.h. ohne führende Nullen) reduziert und in wissenschaftlicher Schreibweise angezeigt. Machen Sie den Versuch; der COMMODORE 16 übersetzt jede Eingabe mühelos! Die wissenschaftliche Schreibweise ist oft nützlich, da der Computer auch größere Zahlen in einer überschaubaren Stellenzahl auswirft. Ein Beispiel: Die Zahl 1 985 054 780 in wissenschaftlicher Schreibweise:

```
1.98505478E+09     Nur eine Stelle links vom Dezimalpunkt.
          !
          Diese Zahl gibt an, um wieviele Stellen der
          Dezimalpunkt in dieser Schreibweise nach links
          verschoben wurde.
```

Zahlen, die kleiner als Ein-Hundertstel ('< 1/100') sind, werden vom COMMODORE 16 ebenfalls in wissenschaftlicher Schreibweise ausgegeben, mit dem Unterschied, daß der Exponent negativ erscheint, d.h. der Dezimalpunkt wird um entsprechend viele Stellen nach r e c h t s verschoben. Zum Beispiel: PRINT .0003359 ergibt 3.359E-04.

Beispiele zur wissenschaftlichen Schreibweise:

20	in wissenschaftlicher Notation	:	2E+01
105000	- " -	:	1.05E+05
.0666	- " -	:	6.66E-02

```
*****************************************
*  5.3  AUSFÜHRUNG VON RECHENOPERATIONEN  *
*****************************************
```

5.3 AUSFÜHRUNG VON RECHENOPERATIONEN

Um eine Rechenoperation auszuführen, ist PRINT und anschließend das mathematische Problem (o h n e Anführungszeichen!) einzugeben:

```
 10 PRINT 1+2, 2-1
 20 PRINT 2*2, 4/2
RUN
 3         1
 4         2
```

Anders als bei Textkonstanten (in Anführungszeichen) deckt sich die Ausgabe am Bildschirm nicht Zeichen für Zeichen mit den Parametern des PRINT-Befehls. Vielmehr errechnet der Computer selbständig die Lösung und druckt nur das Ergebnis aus. Wenn Sie also mit der PRINT-Anweisung rechnen wollen, so brauchen Sie bloß die Anführungszeichen wegzulassen.

Nächstes Experiment:

```
NEW                                  und Taste &lt;Return&gt;
READY.
10 PRINT "2001/2010"                 und Taste &lt;Return&gt;
20 PRINT 2001 - 17                   und Taste &lt;Return&gt;
RUN                                  und Taste &lt;Return&gt;
2001/2010
 1984         Die Leerstelle links von '1984' steht für
              das Vorzeichen, das bei PLUS entfällt.
READY.
```

Die PRINT-Anweisung in Zeile 10 befindet sich zwischen Anführungszeichen, daher wird der dazwischen geschriebene Teil genauso am Bildschirm wiedergegeben. Es wird hierbei auch, im Gegensatz zu Zeile 20, keine Stelle für das Vorzeichen freigelassen. Im nächsten Beispiel ändern Sie Zeile 10 wie folgt und löschen Zeile 20:

```
10 PRINT "2001/2010=";2001/2010      Den Strichpunkt (Semikolon)
                                     nicht vergessen!
20                                   und Taste &lt;Return&gt;
RUN                                  und Taste &lt;Return&gt;
2001/2010= .995522388    Die Leerstelle vor dem Dezimalpunkt
                         steht wieder für das Vorzeichen.
```

Sollen Text und Ergebnis der Rechenoperation gemeinsam am Bildschirm erscheinen, so ist die Schreibweise entsprechend obigem Beispiel (Zeile 10) zu wählen.

VORRANGORDNUNG BEI BERECHNUNGEN
****

Es ist möglich (und zulässig), pro Zeile auch mehrere Berechnungen durchzuführen. Ein Beispiel:

```
PRINT 200*50+5          und Taste <Return>
10005               ... haben Sie mit diesem Ergebnis gerechnet?
                        Versuchen Sie als nächstes diese Eingabe:
PRINT 50+5*200          und Taste <Return>
1050                ... erwarteten Sie dieses Ergebnis?
```

Der COMMODORE 16 führt Berechnungen in einer ganz bestimmten Reihenfolge aus (Vorrangordnung der Algebra). Rechenoperationen werden von links nach rechts (in der Zeile) ausgeführt; innerhalb dieser Generalregel werden einige Rechenoperationen vorgezogen. Nachstehend diese Vorrangordnung:

(1.) Prüft der COMMODORE 16 auf Negativ-Zahlen

(2.) Werden sämtliche Exponential-Berechnungen ausgeführt

(3.) Rechnet der COMMODORE 16 alle Multiplikationen und Divisionen (von links nach rechts)

(4.) Rechnet der COMMODORE 16 alle Additionen und Subtraktionen (von links nach rechts)

ANMERKUNG: Im COMMODORE 16 haben Berechnungen in Klammern Priorität vor allen anderen Berechnungen. Auch mehrere Klammern lassen sich verschachteln: 36 * (12 + (A/3)). Die Berechnung der Klammern erfolgt von innen nach außen.

Eine Verbesserung der Übersicht bringt die Verpackung negativer Zahlen in Klammern. Ein Beispiel: 45 * (-5) ist übersichtlicher, obwohl der COMMODORE 16 diese Berechnung auch ohne Klammern richtig interpretiert.

```
**************************************************
*  5.4 WEITERE BEFEHLE ZUM BILDSCHIRMAUSDRUCK  *
**************************************************
```

Sicher ist Ihnen aufgefallen, daß Komma (',') und Semikolon (';') ein unterschiedliches Ausdruckbild am Bildschirm bewirken. Der COMMODORE 16 interpretiert diese Interpunktionszeichen für den Zwischenraum beim Ausdruck.

Der Unterschied ist ähnlich den bereits beschriebenen Auswirkungen der Anführungszeichen bei Rechenoperationen und Text (sogenannten 'TEXT STRINGS') in Verbindung mit dem PRINT-Befehl. Ein Beispiel soll dies verdeutlichen:

```
NEW          und Taste <Return>
READY.
10 PRINT "O","K"
20 PRINT "O";"K"
```

Beachten Sie, daß sich in beiden Programmzeilen die Interpunktionszeichen a u ß e r h a l b der Anführungszeichen (STRINGS) befinden.

```
RUN          und Taste <Return> bewirkt folgendes:
O         K
OK
```

Wie kommt dieser Unterschied im Ausdruck zustande, obwohl sich Zeile 10 und 20 nur im Interpunktionszeichen unterscheiden?

Werden STRINGS (zwischen Anführungszeichen stehende Zeichen) in einer PRINT-Anweisung durch Komma getrennt, so werden die Einzelstrings beim Ausdruck durch mehrere Leerstellen getrennt.

Werden STRINGS (zwischen Anführungszeichen stehende Zeichen) in einer PRINT-Anweisung durch Semikolon getrennt, so werden die Einzelstrings zusammenhängend ausgedruckt.

Wir rekapitulieren: Der Bildschirm des COMMODORE 16 hat 40 Zeichen pro Zeile und jede dieser Zeilen ist (unsichtbar) in vier Druck-Zonen von je 10 Zeichen eingeteilt. Werden nun PRINT-Parameter durch ein Komma getrennt, so druckt der COMMODORE 16 den ersten STRING in die erste Druckzone und den mit Komma getrennten zweiten STRING in die zweite Zone usw. Das Komma wirkt somit wie ein gesetzter Tabulator auf der Schreibmaschine.

```
        DRUCK-ZONE 1                  DRUCKZONE 2
SPALTE:  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
         O                            K
        DRUCK-ZONE 3                  DRUCKZONE 4
SPALTE: 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
```

DRUCKZONENGESTALTUNG BEIM PRINTBEFEHL MIT KOMMA

Umfaßt der PRINT-Befehl mehr als vier durch Kommata getrennte Strings, so setzt der COMMODORE 16 automatisch in der nächsten Zeile fort.

Ein Beispiel:

```
PRINT "A","B","C","D","E","F"   und Taste <Return>
                                bringt folgenden Ausdruck

        1        11        21        31      40      SPALTE

ZEILE 1 A        B         C         D
      2 E        F
```

Trennen Sie die PRINT-Befehle durch Semikolons, dann werden die Druckzonen vom COMMODORE 16 ignoriert, und der Ausdruck erfolgt zusammenhängend.

Ein Beispiel:

```
PRINT "A";"B";"C";"D";"E";"F"   und Taste <Return>
                                bringt folgenden Ausdruck

        1        11        21        31      40      SPALTE

ZEILE 1 ABCDEF
```

Ist der erste zu druckende STRING länger als 9 Zeichen und der zweite Ausdruck mit einem Komma getrennt, dann passiert folgendes:

```
PRINT "ABCDEFGHIJKL","M"        und Taste <Return>
                                bringt folgenden Ausdruck

        ZONE 1   ZONE 2    ZONE 3    ZONE 4
        1        11        21        31      40      SPALTE

ZEILE 1 ABCDEFGHIJKL       M
```

Was geschieht, wenn für die Eingabe einer Programmzeile die 40 Zeichen einer Zeile nicht ausreichen. Ein Beispiel:

```
10 PRINT "DER COMMODORE 16 IST EIN HANDLICHER, KLEINER COMPUTER"
```

Beim Eintippen dieser Programmzeile reichen die 40 Zeichen einer Zeile nicht aus. Schreiben Sie dennoch weiter; beim COMMODORE 16 springt der CURSOR automatisch am Ende einer Zeile an den Anfang der nächsten Zeile. Programmzeilen können auf diese Weise das max. übliche Maß von 80 Zeichen Länge erreichen. Dabei ist zu beachten, daß sich der CURSOR innerhalb einer der beiden Zeilen befindet, wenn - zur Übergabe an den Computer - die Taste <Return> gedrückt wird. Für den Computer ist nicht das Ende der Zeile, sondern das <RETURN>-Signal das entscheidende Kriterium.

Mit 'RUN' und Drücken der Taste <Return> wird dieser String auch ausgedruckt. Der dabei über 40 Zeichen hinausgehende Teil wird automatisch in der darauffolgenden Zeile ausgedruckt. Auch hier empfehlen sich einige Übungsbeispiele - auch mit mehr als 80 Zeichen.

5.5 VARIABLE

Das Beispiel 36*(12+(A/3)) zeigt u.a. eine der wesentlichen Eigenschaften des COMMODORE 16. Den Einsatz eines Buchstabens in einer mathematischen Formel - den Einsatz einer VARIABLEN. Eine VARIABLE stellt stets einen Wert dar:

```
10 A = 3
20 PRINT "ERGEBNIS:";A * 4
RUN
ERGEBNIS: 12
```

und Taste <Return>, dann erscheint am Bildschirm:

Es sind drei VARIABLEN-Typen, die es zu unterscheiden gilt:

VARIABLE	SYMBOL	BESCHREIBUNG	BEISPIEL	WERT-BEISPIELE
FLIESS-KOMMA		Ganze oder Dezimalzahlen	X, AB, T4	23.5, -12, 1.3E+02
INTEGER	%	Ganze Zahlen	X%, A1%	15, 102, -32700
TEXT-STRING	$	Buchstaben, Zahlen und alle anderen Zeichen zwischen Anführungszeichen	X$, MS$	"SUMME:", "DM" "1. TAG", "COMMODORE"

Eine VARIABLE, die als ganze Zahl betrachtet werden soll, wird mit dem Prozentzeichen ('%') gekennzeichnet.

Eine VARIABLE, die Text enthält, wird mit dem Dollarzeichen ('$') gekennzeichnet.

Fehlen diese Zusätze, so interpretiert der COMMODORE 16 die VARIABLE als Fließkomma-Variable, d.h. als reellen Zahlenwert ('REAL'). Die Integer-Variablen stellen eine Untergruppe der Fließkomma-Variablen dar; sie enthalten keinen Dezimalpunkt.

Es ist wichtig, daß immer der richtige Variablen-Typ zur Anwendung kommt. Wollen Sie z.B. ein Textwort einer Integer-Variablen zuordnen, so wird dies nicht funktionieren. Das nachstehende Beispiel wird Ihnen die Unterschiede demonstrieren.

```
10 PRINT "GEBEN SIE EINE ZAHL EIN"
20 INPUT X%
30 PRINT "ICH LESE IHRE ZAHL ALS";X%
40 PRINT "GEHT PRIMA!"
50 END
```

Mit INPUT übernimmt der COMMODORE 16 Werte von außen.

Starten Sie dieses Programm mit 'RUN' und versuchen Sie die nachstehenden Werte (nach jedem Neu-Start) bei Aufforderung einzutippen (Taste <Return> nicht vergessen).

```
.043
10
EIN VIERTEL
```

NUMERISCHE FUNKTIONEN
****

Die Computersprache BASIC des COMMODORE 16 enthält, wie die wissenschaftlichen Rechner auch, eine Vielzahl numerischer Funktionen (wie z.B. SINUS, COSINUS, TANGENS, usw.).

Die Mehrzahl dieser Funktionen wird mit Eingabe des Funktionsnamens und des zu rechnenden Zahlenwertes (in Klammern) berechnet:

FUNCTION(X)

Um zum Beispiel den Sinus einer Variablen zu finden, genügt es zu schreiben:

PRINT SIN(X),

wobei X jeden gewünschten Zahlenwert annehmen kann. Der Winkel ist übrigens bei allen Winkelfunktionen im 'Bogenmaß' anzugeben (siehe bei SIN im LEXIKON).

Numerische Funktionen lassen sich auch in Programmzeilen einbinden, wie das folgende Beispiel zeigt:

```
10 FOR X = 1 to 5
20 PRINT "DIE QUADRATWURZEL AUS";X;"IST"; SQR(X)
30 NEXT X
```

Beachten Sie auch die vollständige Aufzählung aller im COMMODORE 16 implementierten Numerischen Funktionen im BASIC 3.5 Lexikon.

SELBSTDEFINIERTE FUNKTIONEN
****

Einen sehr wirkungsvollen Einsatz der mathematischen Fähigkeiten des COMMODORE 16 stellt die Möglichkeit dar, eigene Funktionen zu definieren. Diese selbst-definierten Funktionen können sehr hilfreich bei umfassenden Berechnungen sein. Das Prinzip besteht darin, daß der Benutzer seine eigene Formel entwickelt und der Computer die Rechenwerte dazu liefert.

Um z.B. die Fläche eines Kreises zu berechnen, definiert der Benutzer folgende Funktion, die in der Folge nur noch die Eingabe der 'X'-Werte benötigt:

```
10 DEF FNF(X)=2*π*X*X
```

Mit dieser Funktion läßt sich für jeden Wert 'X' die Fläche des Kreises mit Radius X errechnen, wobei 'FNF' der Name dieser definierten Funktion ist. Im Anhang finden Sie eine Reihe mathematischer Funktionen, die nicht im BASIC 3.5 enthalten sind, und auf diese Weise dennoch zur Anwendung kommen können.

```
*********************************
*                               *
*  T  E  I  L     6             *
*                               *
*********************************
*                               *
*  G  R  A  F  I  K             *
*                               *
*  U  N  D                      *
*                               *
*  F  A  R  B  E  N             *
*                               *
*********************************
```

- GRAFIKZEICHEN
  ****
- ZEICHENBEWEGUNG (TRICK)
  ****
- STEUERUNG DER FARBEN
  ****
- HOCHAUFLÖSENDE GRAFIK
  ****
- PUNKTE, LINIEN UND ÜBERSCHRIFTEN
  ****
- QUADRATE, KREISE, VIELECKE
  UND MALEREI
  ****

****
* 6.1 GRAFIKZEICHEN *
****

Im Teil 2 dieses Handbuchs wurde bereits besprochen, daß hinter jeder Buchstabentaste zusätzliche Grafikzeichen stehen. Gleiches bei den Tasten <+>, <->, <*>, <@>, <=> und dem Pfund-Zeichen, wodurch sich insgesamt 62 verschiedene Grafikzeichen ergeben. Um diese Grafikzeichen auszudrucken, müssen entweder die Taste <Shift> oder die COMMODORE-Taste <C=> und gleichzeitig eine der oben beschriebenen Tasten - mit dem gewünschten Symbol - niedergedrückt werden.

Befindet sich der COMMODORE 16 nicht im Schreibmaschinen-Modus, so wird mit Taste <Shift> und der Buchstabentaste das r e c h t e Grafikzeichen gedruckt. Bereits dieser Grafiksatz enthält alle Spielkartensymbole, volle und leere Ballsymbole und eine ganze Reihe sich ergänzender Kurven und Linien. Dieser Satz reicht schon für vielfältigste Anwendungen.

Wird die COMMODORE-Taste <C=> in Verbindung mit einer Buchstabentaste gedrückt, so erscheinen die l i n k e n Grafikzeichen. Diese umfassen insgesamt Balken, Quadrate, Linien und Blöcke, um grafische Darstellungen und Diagramme zu zeichnen.

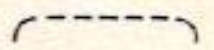
WIE MAN SPIELKARTEN ENTWIRFT
****

Nachstehend wird erklärt, wie vielfältig und sinnvoll sich diese Grafiksymbole einsetzen lassen. Folgen Sie diesen Instruktionen und entwerfen Sie eine richtige Spielkarte, eine 'HERZ-SECHS'.

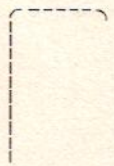
Als erstes wird die CURSOR-Farbe in ROT geändert. Taste <Control> gedrückt halten und die Ziffertaste <3> antippen - und der CURSOR blinkt in Rot.

Nächster Schritt ist die Oberkante der Spielkarte. Da in der Folge nur Grafikzeichen verwendet werden, wird die Taste <Shift> bei allen Zeichen niedergehalten, der CURSOR etwa zehn bis zwölf Positionen vom linken Rand entfernt und aufeinanderfolgend:

1 x Taste <U>, 5 x Taste <C> und 1 x Taste <I>
gedrückt. Nachstehend das Ergebnis:



Nun die linke Spielkartenseite. Die Taste <Shift> wird wiederum bei allen Zeichen gedrückt, und der CURSOR muß in das Feld unter dem linken Bogen (der durch die Taste <U> entstand). Mit der Taste <B> werden ab dieser Position - nach unten - sieben senkrechte Striche gezogen. (Dabei üben Sie auch den Umgang mit dem CURSOR-Kreuz):



Die Unterkante und die rechte Spielkartenseite sind genauso einfach zu zeichnen:

1 x Taste <J>, 5 x Taste <C>, 1 x Taste <K> und 5 x <B>

gedrückt ... und nachstehend das Ergebnis:

```
┌-----┐
|     |
|     |
|     |
|     |
|     |
|     |
|     |
└-----┘
```

Jetzt folgen die Spielkarten-Einträge. Dazu den CURSOR zuerst in die linke, obere Spielkartenecke bewegen und eine '6' eintragen (Taste <6>); und in der rechten, unteren Ecke (eine '6' spiegelbildlich) eine '9'.

Letzter Schritt ist dann die Plazierung der Herz-Symbole, und zwar sechsmal. Diese Einträge müssen symmetrisch auf der Spielkarte verteilt sein - etwa so:

```
┌-----┐
|6    |
| o o |
|     |
| o o |
|     |
| o o |
|    9|
└-----┘
```

Keine allzu gute Karte etwa im Kartenspiel 'Black Jack', aber Sie können ja weiterüben und durch diese Praxis Ihre Fähigkeit erhöhen.

ANMERKUNG: Bei gedrückter Taste <Shift> kann die Taste <Return> ohne Gefahr eines 'SYNTAX ERROR' gedrückt werden, egal was in der Zeile des CURSORS auch geschrieben steht. Der COMMODORE 16 interpretiert in diesem Fall die Taste <Return> nicht der BASIC-Sprache entsprechend als Startbefehl, sondern lediglich als Aufforderung, den CURSOR an den Anfang der nächsten Zeile zu bringen.

Mit Hilfe der Grafik-Tasten läßt sich der Ausdruck wesentlich schöner und interessanter gestalten. Das nächste Beispiel zeigt, wie Worte oder Spalten unterstrichen werden können.

Den CURSOR als erstes in die nächste Zeile (unter das erste zu unterstreichende Zeichen) bewegen. Dann die COMMODORE-Taste <C=> gedrückt halten und die Taste <T> ebenfalls drücken - solange bis die gewünschte Unterstrichlänge erreicht ist. Beide Tasten wieder loslassen.

Zweck dieses Abschnitts war es, die vielfältigen Grafikmöglichkeiten des COMMODORE 16 aufzuzeigen und diese bei der Gestaltung unterschiedlichster Formen und Figuren einzusetzen. Zusätzlich zu den 62 Grafikzeichen steht diese Anzahl nochmals in REVERS-Darstellung zur Verfügung (siehe die Beschreibung der 'Rennstreifen' - Kapitel 4.3). Wie immer macht Übung auch hier den Meister.

```
**********************************
*  6.2  ZEICHENBEWEGUNG (TRICK)  *
**********************************
```

Jeder Film besteht aus einer Vielzahl von Einzelbildern, und jedes Bild unterscheidet sich ein wenig vom vorhergehenden. Der Projektor wirft jedes dieser Bilder nur für eine Sekundenbruchteil auf die Leinwand, und schon kommt das nächste. Durch die Trägheit unseres Auges entsteht für den Betrachter der Eindruck eines bewegten Bildes.

Computer-Bewegung funktioniert genauso. Zuerst bringt der Computer ein Bild, dann wird das Bild ein wenig geändert. Der COMMODORE 16 ist schnell genug (der Bildwechsel muß mindestens 25 mal in der Sekunde erfolgen, um fließende Bewegungen zu erzielen), um den Effekt sich langsam bewegender Objekte auf den Bildschirm - sowohl in Spielen wie in Programmen der Praxis - 'zaubern' zu können.

Programme zur Erstellung solcher Bilder verwenden die PRINT-Anweisung in Verbindung mit den Grafikzeichen. Einfachstes Beispiel eines Bewegungsablaufes sind zwei sich abwechselnde Zeichen.

Ein erstes Beispielprogramm läßt abwechselnd einen Kreis (Taste <Shift> und gleichzeitig Taste <Q> drücken) und ein Herz (Taste <Shift> und gleichzeitig Taste <S> drücken) erscheinen. Mit einiger Phantasie läßt sich dies bereits als 'Herzschlag' interpretieren.

```
10 PRINT "♥ ●":FOR I = 1 TO 200:NEXT
          !  !
          !  Tasten <Shift> & <Q>
          Tasten <Shift> & <Clr/Home>
```

Das Verbindungszeichen '&' bedeutet 'gleichzeitig drücken.'

```
20 PRINT "♡♥":FOR I = 1 TO 200:NEXT
          ! !
          !  Tasten <Shift> & <S>
          Tasten <Shift> & <Clr/Home>
30 GOTO 10
```

Das Verbindungszeichen '&' bedeutet 'gleichzeitig drücken'.

WICHTIG: Für den weiteren Beschreibungsteil wird darauf hingewiesen, daß das Verbindungszeichen '&' den g l e i c h z e i t i g e n Druck der mit diesem Zeichen verbundenen Tasten bedeutet. Die jeweils zuerst genannte Taste ist immer etwas <u>vor</u> der nächsten zu betätigen.

Vor jedem neuen Programm - ob es nun abgespeichert wurde oder nicht - ist es ratsam, 'NEW' einzugeben und die Taste <Return> zu drücken (Speicher wieder in vollem Umfang frei). Jede Programmzeile muß ebenfalls mit der Taste <Return> abgeschlossen werden (damit übernimmt der Computer diese Zeile in den Speicher).

```
10 PRINT "S♠"
          ! !
          !  Tasten <Shift> & <A>
          Taste <Clr/Home>
20 FOR L = 1 TO 100
30 NEXT L
40 PRINT "S♣"
          ! !
          !  Tasten <Shift> & <X>
          Taste <Clr/Home>
50 FOR M = 1 TO 100
60 NEXT M
70 GOTO 10
```

Das Verbindungszeichen '&' bedeutet 'gleichzeitig drücken'.

Das Verbindungszeichen '&' bedeutet 'gleichzeitig drücken'.

Die Grenzen dieser Bewegungsabläufe (nach dem RUN-Befehl) - mit zwei Zeichen - sind klar erkennbar. Gestoppt wird das Programm mit der Taste <Run/Stop>.

Interessantere Effekte lassen sich mit aus mehreren Grafikzeichen zusammengesetzten Bildern erzielen, bei denen im zweiten Bild einige wenige Zeichen geändert werden, die anderen hingegen ihre Position behalten. Dadurch entsteht der Eindruck, als würde sich ein größeres Objekt bewegen. Das nachstehende Beispiel soll dies demonstrieren:

```
 5 SCNCLR
10 PRINT "S \ O /"
          ! ! ! !
          ! ! ! Tasten <Shift> & <N>
          ! ! Tasten <Shift> & <W>
          ! Tasten <Shift> & <M>
          Taste <Clr/Home>

20 PRINT " ▒ "
          ! ! !
          ! ! 1x LEER-Taste <SPACE>
          ! COMMODORE-Taste <C=> & <+>
          1x LEER-Taste <SPACE>

30 PRINT "/ \"
          ! ! !
          ! ! Tasten <Shift> & <M>
          ! 1x LEER-Taste <SPACE>
          Tasten <Shift> & <N>

40 FOR L = 1 TO 100 : NEXT L
50 PRINT "S  O  "
          ! ! ! !
          ! ! ! 1x LEER-Taste <SPACE>
          ! ! Tasten <Shift> & <W>
          ! 1x LEER-Taste <SPACE>
          Taste <Clr/Home>

60 PRINT "-▒-"
          ! ! !
          ! ! COMMODORE-Taste <C=> & <T>
          ! COMMODORE-Taste <C=> & <+>
          COMMODORE-Taste <C=> & <T>

70 PRINT " | | "
          ! ! !
          ! ! COMMODORE-Taste <C=> & <G>
          ! COMMODORE-Taste <C=> & <G>
          1x LEER-Taste <SPACE>

80 FOR L = 1 TO 100 : NEXT L
90 GOTO 10
```

'RUN' eintippen und Taste <Return> drücken.

In den bisher gebrachten Bewegungsbeispielen blieb die bewegte Grafik an einem festen Ort des Bildschirms. In der nächsten Übung wird sich die Figur über den Bildschirm bewegen. Mit Hilfe der TAB-Funktion können Objekte von der linken Bildschirmkante weg bewegt werden (Details zur TAB-Funktion finden Sie im BASIC-Lexikon). Im folgenden Beispiel wird sich eine Schlange über den Bildschirm bewegen.

Und vergessen Sie nicht, daß die mit '&' verbundenen Tasten gleichzeitig gedrückt werden müssen.

```
 5 FOR A = 0 TO 30
10 SCNCLR
20 PRINT TAB(A) "<Shift>&<U> <Shift>&<I> <Shift>&<U> <Shift>&<I>"
30 PRINT TAB(A) "<Shift>&<K> <Shift>&<J> <Shift>&<K> <Shift>&<J>"
40 FOR L = 1 TO 100 : NEXT L
50 SCNCLR
60 PRINT TAB(A+1)"<Shift>&<I> <Shift>&<U> <Shift>&<I> <Shift>&<U>"
70 PRINT TAB(A+1)"<Shift>&<J> <Shift>&<K> <Shift>&<J> <Shift>&<K>"
80 FOR L = 1 TO 100 : NEXT L
90 NEXT A
```

Mit dem Grafikzeichen "Ball" (Tasten <Shift>&<Q>) lassen sich bereits Videospiele auf dem Schirm spielen. Um den Ball zu bewegen, genügt es, ihn zu löschen und an der neuen Position wieder darzustellen - wie nachstehendes Programm zeigt:

```
10 PRINT " <Shift> & <Clr/Home> "
20 PRINT "   ● ▌";
           !  !  !
           !  !  Taste <CRSR/Links>
           !  Tasten <Shift> & <Q>
           1x LEER-Taste <SPACE>

30 FOR L = 1 TO 50 : NEXT L
40 GOTO 20
```

* 6.3 STEUERUNG DER FARBEN *

An jede Stelle des Bildschirms kann eine andere Farbe gesetzt werden. So kann z.B. die Einrahmung des Bildschirms in einer, der Hintergrund in einer zweiten und jeder Buchstabe in einer weiteren eigenen Farbe sein. Wie die einzelnen Buchstabenfarben zu setzen sind, wurde bereits erklärt (Kapitel 2.3). Farbänderungen von Rand und Hintergrund erfolgen über die BASIC-Anweisung COLOR.

Durch die Anweisung COLOR 4,3 und Drücken der Taste <Return> wird z.B. der Bildschirmrand rot. Mit '4' wird in dieser Anweisung der Randbereich und mit der '3' die Farbe Rot angesprochen (Taste <3>).

Bei der Anweisung COLOR 0,7 und der Taste <Return> wechselt die Hintergrundfarbe in die Farbe Blau. Dabei stellt die Null ('0') die Adresse für den Hintergrund dar und die Sieben ('7') für die Farbe Blau (repräsentiert durch die Taste <7>).

Somit können wir festhalten:
die erste Zahl hinter der Anweisung COLOR steht für den Bildschirmbereich, der geändert werden soll. In nachstehender Tabelle wird ein Gesamtüberblick gegeben. Die Bereiche 2 und 3 werden im Kapitel 6.7 Mehrfarbengrafik besprochen.

NUMMERN DER BILDSCHIRMBEREICHE
****

Bereichs-Nr.	Bereichs-Bezeichnung
0	Bildschirm-Hintergrund
1	Buchstaben, Zeichen (Vordergrund)
2	Mehrfarben 1
3	Mehrfarben 2
4	Bildschirm-Rand

Die zweite Zahl in der COLOR-Anweisung selektiert die Farbe des gewählten Bildschirmbereichs. Diese Zahl ist mit den Farbtasten der Tastatur identisch.

FARBNUMMERN
****

Farb-Nr.	Farbe	Farb-Nr.	Farbe
1	Schwarz	9	Orange
2	Weiß	10	Braun
3	Rot	11	Glb/Grün
4	Zyan	12	Rosa
5	Purpur	13	Bla/Grün
6	Grün	14	Hl/Blau
7	Blau	15	Dk/Blau
8	Gelb	16	Hl/Grün

Jede Farbe ist auch noch in ihrer Helligkeit, der 'LUMINANZ' veränderbar. Im Anhang an die Farb-Nummer kann diese LUMINANZ von Null ('0'= dunkel) bis sieben ('7'= hell) variieren. Geben Sie COLOR 4,3,0 ein, und der Bildschirmrand wird sich in dunklem Rot präsentieren. Bei COLOR 4,3,7 erscheint die Einrahmung in hellem Rot.

Zusammenfassend lautet die COLOR-Anweisung:

COLOR (Bereich), (Farbe), (Helligkeit)

Mit dem nachstehenden kurzen Programm werden Ihnen alle 16 Commodore-Farben gezeigt. Zuerst NEW eingeben und Taste <Return> drücken:

```
 5 SCNCLR
10 COLOR 0, 7, 7
20 FOR M=0 TO 7
30 FOR N=1 TO 2
40 FOR L=1 TO 16
50 PRINT "R  ";
          !
          Taste <Control> & <Rvs/On>

60 READ A
70 COLOR 1, A, M
80 PRINT "    ";
          !  !
          !  !
          !  1x LEER-Taste <SPACE>
          1x LEER-Taste <SPACE>

90 NEXT L
100 PRINT
110 RESTORE
120 NEXT N,M
130 COLOR 1, 2, 4
200 DATA 7,14,4,13,6,16,11,8,10,9,3,12,5,15,2,1
```

Wird dieses Programm mit RUN gestartet, so verändert sich die Hintergrundfarbe des Bildschirms in Hellblau, und das Spektrum der 16 Commodorefarben wird im gesamtmöglichen Helligkeitsbereich (0-7) gezeigt. Dabei werden Sie feststellen, daß die Farbe Schwarz im gesamten Helligkeitsspielraum schwarz bleibt.

WICHTIG: Wie für die anderen in diesem Kapitel wiederholten BASIC-Grafik-Kommandos auch, gilt für das BASIC-Wort COLOR, daß es sowohl als Anweisung wie auch als Kommando vorkommen kann. Dabei bleibt es solange unwichtig, dies besonders hervorzuheben (gleiches gilt für die anderen Grafik-Befehle), bis klargestellt ist, welche Formulierung im DIREKT-Modus oder im PROGRAMM-Modus öfter vorkommt.

GRAFIK-MODUS
****

Die bisher gezeigten Grafiksymbole sind Einzelsymbole der Tastatur und stehen in keinem Verhältnis zu den weiteren Möglichkeiten des COMMODORE 16. Im Befehlsvorrat des COMMODORE-16 - BASICs sind Befehle enthalten, die es ermöglichen, Grafiken auch per Programm zu erzeugen. Um diese grafikbezogenen Befehle einsetzen zu können, muß in den GRAFIK-Modus umgeschaltet werden.

Der GRAFIK-Modus kann gleichzeitig als ZEICHNUNGS-Modus verstanden werden, da auch alle für das Zeichnen nötigen Befehle aktiviert sind. Bevor der entsprechende GRAFIK-Modus - per GRAPHIC-Befehl - nicht ausgewählt ist, sind die oben erwähnten Befehle nicht einsetzbar.

Wir unterscheiden drei GRAFIK-Modi: Normaler TEXT, Hochauflösende GRAFIK und Mehrfarbige GRAFIK. Per GRAPHIC-Befehl ist auch die Kombination der Modi möglich, wobei z.B. ein Teil des Bildschirms mit Text beschrieben wird, während auf dem anderen Teil gezeichnet werden kann. Der BASIC-Befehl für diesen Modus lautet GRAPHIC.

Die Syntax des 'Graphic'-Kommandos:

'GRAPHIC (Modus), (Bildschirm löschen Ja, Nein)

Modus	Grafik-Modus
0	Text
1	Hochauflösende Grafik
2	Hochauflösende Grafik und Text
3	Mehrfarben-Grafik
4	Mehrfarben-Grafik und Text

Bildschirm löschen	Modus
0	Bildschirm wird n i c h t gelöscht
1	Bildschirm wird gelöscht

Um vom normalen Grafik-Modus (dem Text-Modus) in den Modus der Hochauflösenden Grafik umzuschalten, ist nur die Anweisung GRAPHIC 2,1 und die Taste <Return> erforderlich. Daraufhin wird der Bildschirm gelöscht, und der CURSOR befindet sich am unteren Bildschirmende. Der COMMODORE 16 hat nun einen in zwei Bereiche geteilten Bildschirm:

> Der obere Bildschirmbereich ist für Grafik vorbereitet, und die letzten fünf Zeilen sind für Texte reserviert. (Werden diese fünf Textzeilen nicht gewünscht, dann lautet der Befehl GRAPHIC 1,1 - dieser Modus hat jedoch den Nachteil, daß auch keiner der eingetippten Befehle zu lesen ist!)

Mit der GRAPHIC-Anweisung ist auch das fortlaufende Hin- und Herschalten zwischen GRAFIK- und TEXT-Modus möglich. Zum TEXT-Modus wird mit der Anweisung GRAPHIC 0 und wieder in den Hochauflösenden GRAFIK-Modus durch GRAPHIC 2 geschaltet, ohne dabei den Bildschirm zu löschen.

Es gibt noch eine Möglichkeit, im Hochauflösenden Grafik-Modus den Bildschirm zu 'clearen'. Mit dem Befehl SCNCLR (SCreeNCLeaR) wird der Bildschirm gelöscht, ohne daß der Grafik-Modus verlassen wird. Nach Umschaltung in den Hochauflösenden Grafik-Modus werden 10 KByte des BASIC-Programmspeicher-Bereichs dafür abgetrennt. Wird ausschließlich mit Grafik gearbeitet, so besteht durch den Befehl GRAPHIC CLR die Möglichkeit, diesen abgetrennten Speicherbereich zurückzugewinnen.

6.4 HOCHAUFLÖSENDE GRAFIK

Die Bildschirmaufteilung des COMMODORE 16 setzt sich aus 25 Zeilen mit je 40 Zeichenpositionen (= insgesamt 1000 Positionen) zusammen. Jedes Zeichen wird aus einer Punktmatrix von 8 Reihen mit je 8 Punkten (= 64 Punkte/Zeichen) gebildet. Pro Bildschirmzeile sind dies 40 Positionen x 8 Punkte pro Matrix-Reihe (= 320 Punkte). Vertikal ergeben sich somit 8 Punkte pro Zeile (= 200 Punkte). In Hochauflösender Grafik sind dies insgesamt 320 x 200 Punkte (= 64 000 Punkte), die beim COMMODORE 16 einzeln angesprochen werden können.

In x-Richtung (Spalten) dürfen Koordinaten-Zahlenwerte von 0 bis 319 vorkommen, in y-Richtung (Zeilen) Werte von 0 bis 199. Dabei entspricht y = 0 der o b e r s t e n Punktreihe!

Im normalen Grafik-Modus ist der Zugriff auf das gesamte Zeichen begrenzt. Dabei mag die Kreation eines Rennstreifens oder einer Spielkarte ein hübsches Beispiel sein, doch die Möglichkeiten bleiben auf die den einzelnen Tasten zugeordneten Symbole beschränkt.

Zwar ergeben sich dabei genügend Möglichkeiten, um die verschiedensten Formen und Figuren zu entwerfen, doch eben nur ein Bruchteil der Möglichkeiten bei Einzelpunktansteuerung. Und diese erweiterte Möglichkeit haben Sie beim COMMODORE 16 im Hochauflösenden Grafik-Modus. Die erzielte Auflösung zeigt sich in der Genauigkeit und den Steuerungsmöglichkeiten beim Zeichnen; in Hochauflösender Grafik lassen sich Punkte, Linien, Kreise und andere Formen durch einfache Befehle zeichnen und wieder löschen.

Es gibt im Hochauflösenden Grafik-Modus (kurz: Hi-Res) eine Einschränkung: Pro 8 x 8 - Zeichen-Position sind nur zwei Farben darstellbar. Dies bedeutet, daß jede Zeichenposition des Bildschirms auf zwei Farben begrenzt ist (Vorder- und Hintergrundfarbe pro Zeichenfeld). Von Zeichenposition zu Zeichenposition kann die Farbe im gesamten Spektrumsbereich wechseln - innerhalb der Zeichenposition nur zwischen zwei Farben.

Im später in diesem Teil beschriebenen Mehrfarben-Grafik-Modus ist es möglich, bis zu vier Farben pro Zeichenposition zu definieren. Dies geht jedoch auf Kosten der Auflösung im Hi-Res-Grafik-Modus.

Das folgende Programmbeispiel zeigt einige der Hi-Res-Möglichkeiten des COMMODORE 16 in Verbindung mit dem Befehl GRAPHIC auf. Löschen Sie dafür mit dem NEW-Befehl den Speicher und tippen dann ein:

```
10 COLOR 0,1
20 GRAPHIC 1,1
30 FOR L=2 TO 16
40 COLOR 1,L,4
50 DRAW 1,0,L*12 TO 319,L*12
60 DRAW 1,L*18,0 TO L*18,199
70 NEXT L
80 FOR L=1 TO 5000 : NEXT
90 COLOR 1,2,7
100 GRAPHIC 0
```

Sie werden an den Übergängen eine gewisse Farbverfälschung feststellen. Diese ergibt sich aus den Grenzen der Hi-Res-Grafik, wenn zu viele Farben auf zu engem Raum zusammentreffen.

```
*****************************************
*  6.5  PUNKTE, LINIEN UND ÜBERSCHRIFTEN  *
*****************************************
```

Geben Sie die Befehlsfolge GRAPHIC 2,1:DRAW 1,0,0 ein und drücken Sie die Taste <Return>. Und beobachten Sie genau die linke, obere Bildschirmecke. Der COMMODORE 16 hat mit dieser Befehlsfolge einen einzigen, schwarzen Punkt dorthin gesetzt. Mit dem Befehl DRAW können Einzelpunkte, Linien oder ganze Formen an jede Stelle des Bildschirms gezeichnet werden. Hier einige der DRAW-Befehle:

Befehl	Ergebnis
DRAW (Farbquelle), (Spalte),(Zeile)	PUNKT
DRAW (Farbquelle), (Spalte),(Zeile)TO(Spalte),(Zeile)	LINIE
DRAW (Farbquelle)TO(Spalte),(Zeile)	ZEICHNET LINIE AB DEM LETZTEN PUNKT

Farbquelle Null ('0') steht für den Farb h i n t e r grund des Zeichens, Eins ('1') für die Farbe des Vordergrunds. Wird ein Punkt in der Hintergrundfarbe gesetzt, ist dies gleichbedeutend mit Löschung der Vordergrundfarbe an dieser Punktstelle.

In der Anweisung DRAW kann die erste Zahl entweder 1 (zeichne einen Punkt) oder 0 (lösche einen Punkt) sein. Die nächsten beiden Zahlen geben die Spalten- und Zeilenposition des Punktes an. Soll ein Punkt in Spalte 17, Zeile 20 gesetzt werden, dann lautet der Befehl DRAW 1,17,20. Gelöscht wird dieser Punkt wieder mit DRAW 0,17,20.

Mit der DRAW-Anweisung lassen sich auch Linien zwischen Punkten ziehen. Durch Hinzunahme des Befehls TO und der zweiten Punktposition: DRAW 1,1,1 TO 100,100. Damit wird eine Linie von dem Punkt in Position 1,1 nach dem Punkt in Position 100,100 gezogen. Diese Linie wird wieder gelöscht, wenn der Befehl lautet: DRAW 0,1,1 TO 100,100.

Wird diese Zeichentechnik auch im mathematischen (oder geometrischen) Bereich eingesetzt, so ist ein gewisser Umdenkprozeß notwendig. Im Gegensatz zum herkömmlichen Koordinatensystem (bei dem die Position 0,0 entweder in der Bildmitte oder links unten zu finden ist) beginnt der COMMODORE 16 in der linken oberen Ecke mit seiner Positionszählung. Bei einiger Übung wird dies für Sie jedoch kein Problem darstellen.

Ist erst einmal der erste Punkt oder die erste Linie am Bildschirm gezeichnet, so kann kann die Zeichnung von da fortgesetzt werden: DRAW 1 TO 150,50 bewirkt eine Linie - vom letzten Punkt aus - zur Position Spalte 150, Zeile 50. Werden in einem Programm eine Vielzahl DRAW TO-Befehle verwendet, so empfiehlt sich der Einsatz eines weiteren Befehls: LOCATE.

Mit dem Befehl LOCATE wird der Anfangspunkt an eine Bildschirmposition fixiert und damit die unverzügliche Rückkehr zu diesem Punkt erreicht: z.B. LOCATE 100,100.

NEW und Taste <Return>

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 1,1
40 LOCATE 0,100
50 FOR X=1 TO 319
60 Y = INT (100+99*SIN(X/20))
70 DRAW 1 TO X,Y
80 NEXT X
90 FOR L=1 TO 5000
100 NEXT L
110 GRAPHIC 0
```

Geben Sie nach der Erprobung des obigen Programms k e i n NEW ein. Um eine weitere Version laufen zu lassen, ändern Sie lediglich Zeile 70 in:

```
70 DRAW 1,X,Y
```

Mit dieser Änderung wird das Programm nicht mit Linien sondern mit Punkten zeichnen.

DIE CHAR-ANWEISUNG
****

Grafische Darstellungen werden übersichtlicher, wenn sie zusätzliche Beschriftung erhalten. Mit der Anweisung CHAR kann Hi-Res-Grafik mit Text gemischt werden. Die Anweisung CHAR 1,0,5,"ZEITACHSE" schreibt den Text 'ZEITACHSE' linksbündig in die 6.(!) Zeile des Bildschirms. Die erste Zahl in der CHAR-Anweisung entscheidet wiederum über Darstellen (= 1) oder Löschen (=0). Die folgenden zwei Zahlen bestimmen Spalte und Zeile, ab der der Text erscheint.

Belassen Sie eines der beiden letzten Programme im Computer; geben Sie weder NEW noch <Return> ein, sondern fügen Sie nachstehende Programmzeilen hinzu:

```
81 CHAR 1,0,0,"GRAFISCHE DARSTELLUNG DER":CHAR 1,0,1,"FORMEL"
82 CHAR 1,0,2,"Y = SIN(X)"
83 DRAW 1,0,100 TO 319,100,189,0 TO 189,199
84 CHAR 1,0,12,"X-ACHSE": CHAR 1,22,0,"Y"
85 CHAR 1,22,2,"A": CHAR 1,22,3,"C"
86 CHAR 1,22,4,"H": CHAR 1,22,5,"S"
87 CHAR 1,22,6,"E"
```

6.6 QUADRATE, KREISE, VIELECKE UND MALEREI

RECHTECKE ZEICHNEN

Unter Verwendung des DRAW-Befehls lassen sich Bilder durch Aneinanderreihen vieler Punkte und Linien zeichnen (= plotten). Ein Quadrat läßt sich mit folgendem Befehl zeichnen: DRAW 1,0,0 TO 100,0 TO 100,100 TO 0,100 TO 0,0. Dadurch werden die vier Eckpunkte des Quadrats mit vier Linien verbunden (wobei jeder Eckpunkt des Quadrats geplottet wird). Aber es geht noch einfacher mit dem Befehl BOX.

DIE BOX-ANWEISUNG

Dank der COMMODORE-16-Anweisung BOX wird das Zeichnen von Quadraten oder rechteckigen Figuren noch einfacher. Mit der Anweisung BOX werden die zwei gegenüberliegenden Eckpunkte festgelegt. Das obige Beispiel kann demnach auch so gezeichnet werden: BOX 1,0,0,100,100.

Die erste Zahl bestimmt wieder Zeichnen oder Löschen, und die anschließenden vier Zahlen stellen die Koordinaten der gegenüberliegenden Quadrat-Eckpunkte dar. Für die linke, obere Ecke 0,0 und für die rechte, untere Ecke 100,100 (nahe der Bildschirmmitte).

Mit Hilfe der BOX-Anweisung läßt sich jedes Rechteck zeichnen, indem nur die Eckpunkte geändert werden. Auch gedreht kann diese 'BOX' werden. Dafür ist nach der letzten Koordinate noch der Drehwinkel (in Grad) einzugeben: BOX 1,50,50,100,100,45. Mit diesem Befehl wird die 'BOX' um 45 Grad im Uhrzeigersinn gedreht dargestellt, so daß die Form eines Rhombus entsteht.

Statt der Umrißzeichnung läßt sich auch eine volle Fläche darstellen. Dies geschieht mit einer weiteren Ergänzung: nach der Winkelangabe folgt ein zweites Komma und die Zahl '1'. Eine Quadratfläche etwa in der Bildschirmmitte wird dann so erzeugt: BOX 1,100,50,220,150,,1.

WICHTIG: Auch wenn die Fläche nicht gedreht werden soll, ist dieses zweite Komma wichtig. Der Computer interpretiert dann den fehlenden Winkelwert als Unterlassung (DEFAULT = 0), und deutet dies in der Folge als eine weitere Instruktion. Fehlt das zweite Komma, so wird die Zahl '1' am Befehlsende als Drehwinkel von 1 Grad ausgelegt und die Figur um diesen Wert gedreht - und nicht gefüllt.

Einige typische Beispiele von BOX-Anweisungen:

Befehlsfolge	Ergebnis
BOX ein, Spalte 1, Zeile 1, Spalte 2, Zeile 2	Umriß
BOX ein, Spte 1, Zle 1, Spte 2, Zle 2, Winkel	Gedreht
BOX ein, Spte 1, Zle 1, Spte 2, Zle 2,,Füllen	Fläche
BOX aus, Spte 1, Zle 1, Spte 2, Zle 2, Winkel, Füllen	Gelöscht

Spalte 1, Zeile 1, usw. sind Bildschirmpositionen der gewünschten Eckpunkte. Spalte und Zeile 1 ist die obere, linke Ecke der 'BOX', und Spalte und Zeile 2 die untere, rechte Ecke.

Nachstehendes Programm verdeutlicht die BOX-Anweisung (Zeile 60):

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 1,1
40 A = RND(1)*20+10
50 FOR L=0 TO 359 STEP A
60 BOX 1,100,50,220,150,L
70 NEXT L
80 FOR L=1 TO 2000: NEXT L
90 GRAPHIC 0,1
```

Mit Eingabe von RUN und Taste <Return> wird das Programm gestartet.

KREISE ZEICHNEN
****

Der COMMODORE 16 kennt auch einen Befehl, um Kreise zeichnen zu können. Ähnlich der Anweisung BOX läßt sich die Form in ein Oval oder eine Ellipse verändern. Die nicht kreisförmigen Figuren können auch gedreht werden. Auch Kreissegmente lassen sich auf diese Weise zeichnen.

DIE CIRCLE-ANWEISUNG

Die Standardform der Anweisung CIRCLE lautet:

Befehl	Ergebnis
CIRCLE ein, Mittelpkt Spalte, Mittelpkt Zeile, Radius	KREIS/OVAL
CIRCLE ein, Mtlpt Splte, Mtlpt Zle, Breite, Höhe	KREIS/OVAL
CIRCLE ein, Mtlpt Splte, Mtlpt Zle, Brte, Hhe, Afg, End	KREIS-SEGM
CIRCLE ein, Mtlpt Splte, Mtlpt Zle, Brte, Hhe,,,Winkel	KREIS/OVAL
CIRCLE ein, Mtlpt Splte, Mtlpt Zle, Brte, Hhe,,,,PktWkl	VIEL-ECK

Mit dem folgenden Befehl zeichnen Sie einen Kreis in Bildschirmmitte: CIRCLE 1,160,100,50. Die Koordinaten des Kreismittelpunktes liegen in Spalte 160 und Zeile 100, bei einem Radius von 50. Auf manchen Fernsehern erscheint das Ergebnis möglicherweise nicht als Kreis sondern oval, weil die Bildschirmpunkte (DOTS) höher als breit sind. Um einen 'echten' Kreis zu zeichnen, muß dem Computer in diesem Fall eine Zusatzzahl für die Höhe gegeben werden, die etwas geringer als die der Breite ist. In diesem Fall wird die vierte Zahl nämlich nicht als Radius sondern als Breite erkannt. CIRCLE 1,160,100,50,42.

Der COMMODORE 16 kann mit dem Befehl CIRCLE auch Quadrate, Dreiecke oder andere Vielecke zeichnen. Die Zusatzangabe hierfür ist das Winkelmaß auf einem Kreis zwischen den Eckpunkten: CIRCLE 1,160,100,50,,,,120. Mit diesem Befehl entsteht ein Dreieck, dessen Winkel je 120 Grad betragen.

Erinnern wir uns, daß im CIRCLE-Befehl die anstelle der (weggelassenen) Zahlenwerte eingesetzten Kommata den Computer veranlassen, stattdessen Standardwerte einzusetzen. Die Formel für den Mittelpunktswinkel eines regelmäßigen Vielecks mit n Seiten lautet: 360/n.

Nachfolgend ein kurzes Programm, um Vielecke zu zeichnen:

```
 5 DO
10 GRAPHIC 2,1
20 INPUT "WIE VIELE SEITEN"; A
30 IF A<2 OR A>100 THEN PRINT "BITTE NICHT UEBERTREIBEN !": GOTO20
40 CIRCLE 1,160,80,40,40,,,,360/A
50 INPUT "WEITER J/N";C$
60 LOOP UNTIL C$="N"
70 GRAPHIC 0:END
```

Es steht Ihnen frei, statt eines Kreises auch nur ein Kreissegment zu zeichnen. Die CIRCLE-Anweisung akzeptiert auch Anfangs- und Endpunkte in Grad, wenn sie anschließend an die Höhenzahl eingegeben werden. Mit dem Befehl: CIRCLE 1,160,100,50,42,90,180 wird nur das rechte untere Segment eines Kreises gezeichnet.

Ein Oval läßt sich drehen, wenn hinter dem eigentlichen CIRCLE-Befehl der Winkel (im Uhrzeigersinn betrachtet) der Drehung eingegeben wird: CIRCLE 1,160,100,100,20,,,30.

Und nun ein Programm, in dem die CIRCLE-Anweisung für einen interessanten Effekt genutzt wird. Dazu muß jedoch vorher der Speicher mit dem Befehl NEW (und Taste <Return>) gelöscht werden.

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 1,1
40 A = RND(1)*20+10
50 FOR L=0 TO 359 STEP A
60 CIRCLE 1,160,100,80,40,,,L
70 NEXT L
80 FOR L=1 TO 2000: NEXT L
90 GRAPHIC 0,1
```

DIE PAINT-ANWEISUNG
****

Besteht der Wunsch, von einem Kreis oder einer sonstigen geometrischen Figur nicht nur die Umrisse zu zeichnen, sondern die Fläche auch auszumalen, dann steht dafür die Anweisung PAINT zur Verfügung. Die PAINT-Anweisung füllt jede geschlossene Fläche bis an die Umrandung aus. Fehlen diese Linien, dann erfolgt die Füllung bis zum Bildschirmrand.

Die BOX-Anweisung kennt ebenfalls ein Füllkommando, mit dem Kästchen und Rechtecke eingefärbt werden können. Für alle sonstigen Figuren, die auf dem Bildschirm mit anderen Befehlen nicht gefüllt werden können, steht die Anweisung PAINT zur Verfügung. Sie füllt diejenige geschlossene Fläche aus, in der sich die angegebenen Koordinaten befinden.

Um zu sehen, wie die Anweisung PAINT u.a. einsetzbar ist, geben Sie zum vorhergegangenen Programm noch diese Zeile ein:

```
75 PAINT 1,160,100
```

****
* 6.7 MEHRFARBIGE GRAFIK *
****

Mit der Hi-Res-Grafik des COMMODORE 16 läßt sich jeder einzelne Punkt - auch 'PIXEL' genannt - auf dem Bildschirm ansprechen und steuern. Wir konnten aber auch feststellen, daß die Farben an den Stoßstellen sich verwaschen. Die meisten Hi-Res-Programme lassen überhaupt nur den Einsatz von zwei Farben zu.

Werden mehr Farben gewünscht, so verfügt der COMMODORE 16 über einen MEHRFARBEN-Befehl. Im Gegensatz zur Hi-Res-Grafik stehen im MEHR-FARBEN-Modus nur die Hälfte der Punkte pro Zeile zur Verfügung. Die Punkte selbst sind doppelt so breit wie im Hi-Res-Modus; daher nur 160 Punkte pro Punktzeile - bei gleichbleibend 200 Punktzeilen. Für die MEHRFARBEN-Darstellung muß also eine etwas geringere Auflösung in Kauf genommen werden.

Zum Verständnis des MEHRFARBEN-Befehls betrachten wir nochmals den GRAFIK-Modus (Kapitel 6.3). Hier wurden bereits beide Varianten erwähnt: MEHRFARBEN-Darstellung ohne Text benötigt den Befehl GRAPHIC 3 und, mit fünf Zeilen Text kombiniert, den Befehl GRAPHIC 4.

Ebenfalls im Kapitel 6.3 wurden beim Bereich des COLOR-Befehls die Mehrfarben 1 und 2 (COLOR 2 und COLOR 3) erwähnt. Mit diesen Bereichen stehen zwei weitere Farben zur Verfügung. Jede dieser drei Farben kann eingesetzt werden: 1. die Text-Farbe, 2. die eine Extra-Farbe und 3. die andere Extra-Farbe. Diese Farben verwaschen nicht an den Stoßstellen des Bildschirms, wie dies in einigen beschriebenen Programmen des Hi-Res Modus der Fall war.

Mit dem nächsten Programm-Beispiel setzen Sie MEHRFARB-Grafik ein, wobei ein gewisser 'Leuchtreklame'-Effekt auftreten wird:

NEW

eingeben und Taste <Return>.

Mit der <STOP>-Taste kann das Programm beendet werden. Durch Zeile 40 wird in diesem Fall nach Zeile 200 gesprungen.

```
10 COLOR 0,1
20 GRAPHIC 3,1
30 COLOR 3,1
40 TRAP 200
50 DRAW 3,10,10 TO 10,100: DRAW 3,10,55  TO 30, 55
60 DRAW 3,30,10 TO 30,100: DRAW 3,50,10  TO 80, 10
70 DRAW 3,65,10 TO 65,100: DRAW 3,50,100 TO 80,100
80 FOR L=0 TO 7
90 COLOR 3,2,L
100 FOR M=1 TO 100: NEXT M
110 NEXT L
120 COLOR 3,1
130 FOR M=1 TO 100: NEXT M
140 GOTO 80
200 GRAPHIC 0: COLOR 1,1,7
```

Den Mehrfarben-Bereich 3 (Befehl COLOR 3) verfügt über eine weitere Fähigkeit, die keiner der anderen Bereiche besitzt. Haben Sie einmal Bildschirmbereiche mit dem Befehl COLOR 3 gezeichnet, dann kann diese Farbe an jeder Bildschirmstelle, auf der sie auftaucht, mit der COLOR-Anweisung verändert werden.

Legen Sie z.B. die Farbe mit dem Befehl COLOR 3,5 fest und zeichnen in der Folge mit dieser Farbe, so erscheint die Grafik in Purpur. Wechseln Sie nun per Befehl COLOR 3,6 die Farbe, so wechseln schlagartig alle Purpur eingefärbten Flächen in die neue Farbe Grün. Mit keinem der anderen Bereiche ist dies möglich.

```
*******************************
*                             *
*  T E I L   7                *
*                             *
*******************************
*                             *
*  T Ö N E                    *
*                             *
*  U N D                      *
*                             *
*  M U S I K                  *
*                             *
*******************************
```

* EINLEITUNG
  ****
* LAUTSTÄRKEREGELUNG (VOL)
  ****
* TONERZEUGUNG (SOUND)
  ****
* KLANGEFFEKTE ERZEUGEN
  ****

7.1 EINLEITUNG

Zur Einleitung ein kleines Musikprogramm. Nachdem Sie es in den COMMODORE 16 eingetippt und mit 'RUN' gestartet haben, wird ein Fragezeichen am Bildschirm erscheinen. In diese INPUT-Abfrage können Sie, ganz nach Ihrem Gutdünken, die Zahlen von 1 bis 1015 eingeben - und das Programm per Taste <Return> weiterlaufen lassen. Beendet wird das Programm, indem Sie den Wert Null ('0') eingeben.

```
10 VOL 8
20 DO
30 INPUT X
35 IF X>1015 OR X<0 THEN PRINT "BITTE '0' bis '1015'": GOTO 30
40 SOUND 1, X, 10
50 LOOP UNTIL X=0
```

Einzelne Noten werden auf dem COMMODORE 16 so erzeugt:

1. VOL 8 eintippen und Taste <Return> drücken
2. SOUND 1,266,50 eintippen und Taste <Return> drücken.

Der programmierte Ton erklingt etwa für eine Sekunde. Sehen Sie diese Note als den allerersten Schritt zu einer Symphonie, nach dem Motto: erste Note im Computer, 3 500 weitere noch zu komponieren.

Wenn Sie nichts hörten, dann ist der Lautstärkeregler des Fernsehgeräts oder Monitors nicht genügend weit aufgedreht. Holen Sie dies nach und starten Sie den Zweizeiler nochmals. Er umfaßt alles, um mit dem COMMODORE 16 Musik zu machen. Beide Befehle sind leicht zu verstehen und noch einfacher anzuwenden.

```
**********************************
*  7.2  LAUTSTÄRKEREGELUNG (VOL)  *
**********************************
```

Mit dem Befehl VOL (= VOLume) wird die Lautstärke jedes vom COMMODORE 16 erzeugten Tones gesteuert. Der Befehl funktioniert ähnlich dem Lautstärkeknopf: Folgt eine Null ('0') dem VOL-Befehl (VOL 0), dann bewirkt dies Lautstärke 0. Maximum wird mit der Zahl 8 erreicht, und die Zwischenwerte liegen von 1 bis 7.

Setzen Sie zur Übung in das vorangegangene Beispiel unterschiedliche Werte beim VOL-Befehl ein. Je größer die Zahl, desto lauter der Ton.

```
******************************
*  7.3  TONERZEUGUNG (SOUND)  *
******************************
```

Im SOUND-Befehl sind alle übrigen zur Tonerzeugung erforderlichen Informationen für den COMMODORE 16 enthalten. Diese drei Zahlen beschreiben eine Note wie folgt:

SOUND (Stimmen-Nr.), (Noten-Wert), (Klang-Dauer)

Die erste Zahl im SOUND-Befehl betrifft die Stimme. Man kann Stimme 1, 2 oder 3 wählen. Im COMMODORE 16 sind zwei unabhängige Tongeneratoren eingebaut. Tongenerator 1 erzeugt nur Töne, während Tongenerator 2 entweder Töne oder Rauschen erzeugen kann.

Stimme 1: Tongenerator 1 erzeugt Töne. Wird mit SOUND 1 aufgerufen.

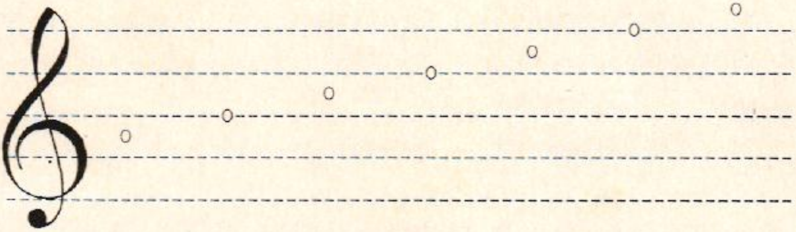
Stimme 2: Tongenerator 2 erzeugt Töne. Verhält sich wie Stimme 1, wenn die Anweisung SOUND 2 lautet.

Stimme 3: Wünschen Sie statt eines Tons Geräusche, wie z.B. Regen oder Donnergrollen, dann tippen Sie SOUND 3, Tongenerator 2 erzeugt Rauschen.

Die zweite Zahl in der SOUND-Anweisung steuert den Notenwert (die Tonhöhe der Note). Dieser Wert kann sinnvollerweise zwischen Null ('0') und 1015 liegen. Der COMMODORE 16 wird dadurch in der Tonhöhe gesteuert. Je höher die Zahl, umso höher der Ton. Die höchsten Werte (ab 1015) sind für das menschliche Ohr kaum noch hörbar.

WICHTIG: Bei Stimme 3 ergeben Werte zwischen 600 und 940 eine Art weißes Rauschen. Um interessante Klangeffekte zu erzeugen, können Register-Werte auch außerhalb dieses Bereichs eingesetzt werden.

Einige Notenbeispiele zum besseren Verständnis. Eine vollständige Tabelle finden Sie im Anhang des Handbuchs.



NOTE:	a	h	c	d	e	f	g
WERT:	770	798	810	834	854	864	881
FREQUENZ: (in Hz)	440.4	494.8	522.7	588.7	658	699	782.2

Versuchen Sie folgendes Programm:

```
NEW                         eingeben und Taste <Return>.

10 VOL 7    <------------ Lautstärke '7'
20 X = 0
30 DO
40 SOUND 1,X,5  <------- Stimme, Notenwert u. Klangdauer
50 X = X + 5
60 LOOP UNTIL X = 1015
70 VOL 0    <------------ Lautstärke '0' (= AUS)
80 END
```

Dieses Programm wird Sie von den musikalischen Fähigkeiten des COMMODORE 16 sicherlich überzeugt haben.

Die dritte Zahl im SOUND-Befehl steuert die Klangdauer (duration) der Note in einem Bereich von 0 bis 65 535 (mal 1/50 Sekunden). Das bedeutet, daß eine Klangdauer von 50 etwa einer Sekunde entspricht. Über den Daumen gepeilt bedeutet eine umso größere Zahl eine entsprechend längere Klangdauer. So erreichen Sie mit der Zahl 65535 eine Klangdauer von über 20 Minuten. Mit dem Wert Null ('0') können Sie jeden Ton ausschalten.

7.4 KLANGEFFEKTE ERZEUGEN

Klangeffekte lassen sich sowohl mit Musiktönen wie mit Geräusch erzeugen. Die Verbindung einfacher BASIC-Programme mit SOUND-Befehlen kann ungewöhnliche und unterhaltsame Effekte bringen. So kann (wie im nachstehenden Beispiel gezeigt) die FOR ... NEXT ... STEP ...-Schleife geschickt für Klangeffekte genützt werden.

Mit diesem Befehl wird eine Schleife gebildet; und jedesmal wenn der Computer den Befehl FOR liest, wird die Zählervariable (in unserem Beispiel 'S') verändert. Mit Ausführung des Befehls NEXT wird zum Befehl FOR zurückgesprungen. In der verwendeten FOR...NEXT-Schleife steht ein negativer STEP Befehl, um von der höheren Zahl herunterzuzählen (jedesmal um 25).

```
10 VOL 8                          Lautstärke auf 8.
20 FOR S=1000 TO 700 STEP -25     Bildet Schleife, um 25 ab-
                                  wärts zählend.
30 SOUND 1, S, 1
40 NEXT S
```

Geben Sie den Befehl RUN in Verbindung mit der Taste <Return> ein und lauschen Sie den Klangeffekten. Programmbestimmend ist die Zeile 20, in der im Zahlenbereich 1000 bis 700 die gesamte Skala in 25er Schritten hinuntergespielt wird. Zeile 30 entscheidet über die Dauer jedes Tons - in diesem Fall von je 1/50 Sekunde. Experimentieren Sie mit den unterschiedlichsten Zahlenwerten bei Klang und Tondauer, womit auch Ihnen einige bemerkenswerte Tonkombinationen gelingen werden.

Wird bei der Stimm-Auswahl der Wert 3 (SOUND 3,.,.) gewählt, dann wird vom COMMODORE 16 Rauschen erzeugt. Damit lassen sich wie bei der Tonerzeugung die unterschiedlichsten Klangeffekte erzielen. Nachstehendes Programm benützt den SOUND 3,.,.-Befehl (Stimme 3), um das Geräusch eines Sturmes nachzubilden.

```
NEW                              eingeben und Taste <Return>.

10 VOL 2   <-------------- Lautstärke '2'
20 R=INT(RND(0)*10)+1  <-- Zufallszahl 'R'
30 FOR X=1 TO R
40 SOUND 3, 600+30*X, 10
50 NEXT X
60 FOR X= R TO 1 STEP -1
70 SOUND 3, 600+30*X, 10
80 NEXT X
90 T=INT(RND(0)*100)+30
100 SOUND 3, 600, T
110 GOTO 20
```

Die Zeilen 30 und 60 bilden je eine FOR...NEXT-Schleife für den Noten-Wert (Frequenz), wobei die erste ansteigende und die zweite absteigende Frequenzwerte - jeweils basierend auf den Zufallszahlen der Zeile 20 - erzeugt. Wichtig für unser Programmbeispiel ist die wechselnde Tonhöhe, da Gewitterstürme mit peitschenden Sturmböen daherfegen.

In den Zeilen 40 und 70 werden per SOUND-Befehl die Geräusche erzeugt, und in den Zeilen 90 und 100 werden mit zufälligen Verzögerungswerten die absolut unregelmäßigen Zeitabstände zwischen dem Heulen eines Gewittersturms nachempfunden.

Außerdem wird in Zeile 90 eine weitere Zufallszahl erzeugt, mit dem die Dauer des SOUND-Befehls in Zeile 100 bestimmt wird. Dieser SOUND-Befehl bleibt auf konstanter Tonhöhe und bildet als Kontrast zum Auf- und Abschwellen des Sturms ein konstantes Hintergrundgeräusch.

Das klingt natürlich alles recht kompliziert und ausschließlich für Programmierprofis machbar. In Wirklichkeit ist nur ein wenig Experimentierfreude notwendig, wobei unterschiedliche Werte eingesetzt werden und man beobachtet, was dabei herauskommt. Geräusche zu erzeugen ist eine Herausforderung, bei der versucht wird, die richtigen Werte für einen gewünschten Effekt zu finden.

WIR MACHEN MUSIK
****

Auch wenn Sie (noch) nicht alles verstehen, was in allen diesen Programmen abläuft, so tippen Sie die Programme dennoch ein und beobachten ihre Wirkung.

Das folgende Programm simuliert über die Tasten <1> bis <8> eine Klaviertastatur.

```
5 SCNCLR
7 PRINT"SPIELEN SIE MIT DEN TASTEN 1..8!"
8 PRINT"ENDE:LEERTASTE"
10 FOR X=1 TO 8: READ N(X): NEXT X
20 VOL 7
30 DO
40 GET A$: IF A$="" THEN 40
50 A=ASC(A$): IF A<49 OR A>56 THEN 90
60 N=A-48
70 SOUND 1, N(N), 5
80 COLOR 0, N, 3
90 LOOP UNTIL A=32
100 VOL 0: COLOR 4, 2, 7
110 DATA 169, 262, 345, 383, 453, 516, 571, 596
```

Spielen Sie mit den Tasten <1> bis <8> die Noten. Mit jeder anderen Note wechselt auch die Bildschirm-Randfarbe. Wollen Sie beenden, so genügt ein Druck auf die LEER-Taste.

Nachdem der COMMODORE 16 auch als Klavier zur Verfügung steht, wollen Sie vielleicht ein bekanntes Liedchen spielen. Ein bekanntes, ja klassisches Beispiel ist mit den zu drückenden Zifferntasten nachstehend abgedruckt:

Tasten der Reihe nach (im Takt) drücken:

1 1 5 5 6 6 5

4 4 3 3 2 2 1

5 5 4 4 3 3 2

5 5 4 4 3 3 2

1 1 5 5 6 6 5

4 4 3 3 2 2 1

Die einzelnen zu spielenden Noten können auch in DATA-Werten (z.B. für die Variablen X und Y) abgelegt und durch das Programm aufgerufen (READ) werden, wobei sich ihr Wert bei jeder Schleife ändert. Im nächsten Programm sind diese DATA-Werte paarweise zu sehen. Der erste Wert bildet stets den Notenwert für den SOUND-Befehl und der zweite Wert erzeugt im SOUND-Befehl die Klangdauer.

'Alle Vögel sind schon da'

```
10 VOL8
20 DO
30 READX,Y
40 SOUND1,X,Y
50 LOOP UNTIL X=0
60 END
100 DATA 596,45,685,15,739,30,810,30
110 DATA 770,30,810,15,770,15,739,60
120 DATA 704,45,739,15,685,30,596,30
130 DATA 643,60,596,30
140 DATA 0,1
```

COMMODORE 16 - DIE MUSIKMASCHINE
****

Das letzte Programm ist etwas länger. Es zeigt den COMMODORE 16 als die 'große' Musikmaschine. Jede Taste zwischen <1> bis <9>, die Sie drücken, erscheint auch als geschriebene Note in den Notenzeilen.

```
5 GOSUB 1000
6 FOR X=1 TO 9: READ N(X): NEXT X
8 CHAR 1, 8, 1,"DIE GROSSE MUSIKMASCHINE"
10 VOL 7
20 DO
30 GET A$: IF A$="" THEN 30      zwischen den Anführungszeichen
                                         k e i n e n  Zwischenraum!
35 A=ASC(A$): IF A<49 OR A>57 THEN 50
36 N= A - 48
40 SOUND 1, N(N), 4
45 GSHAPE N$, 150, 8 * (6+(9-N)), 4
46 FOR Z=1 TO 50: NEXT Z
47 GSHAPE N$, 150, 8 * (6+(9-N)), 4
50 LOOP UNTIL A=32
55 VOL 0: GRAPHIC 0: SCNCLR
60 END
100 DATA 345, 383, 453, 516, 571, 596, 643, 685, 704
1000 GRAPHIC 1,1
1010 FOR Y=60 TO 124 STEP 16
1020 DRAW 1, 100, y TO 200, Y
1030 NEXT Y
1040 A$="FEDCHAGFE"
1050 FOR X=1 TO 9: C=13
1060 IF INT(X/2)= X/2 THEN C=14
1070 CHAR 1, C, X+6, MID$ (A$, X, 1), 0
1075 CHAR 1, C+10, X+6, RIGHT$ (STR$ (10-X), 1)
1080 NEXT X
1090 FOR X=1 TO 8: FOR Y=11 TO 16: DRAW 1, X, Y: NEXT Y, X
1100 Y=1: X=8: DRAW 1, 8, 16 TO X, Y
1110 SSHAPE N$, 1, 1, 8, 16
1120 GSHAPE N$, 1, 1, 4
1130 RETURN
```

Wie Sie feststellen konnten, können Musik und Töne sowohl zur Programmuntermalung genutzt werden oder auch Hauptpunkt eines Programms sein. Die gebrachten Beispiele dieses Kapitels konnten Ihnen nur einen Vorgeschmack der Möglichkeiten des COMMODORE 16 bieten. Scheuen Sie nicht neue Töne und Geräusche zu erproben, und komponieren Sie Ihr eigenes Meisterstück.

Damit schließt der Einführungsteil des COMMODORE 16 Handbuchs. Aufgabe dieses Teils war es, Ihnen einen Vorgeschmack dessen zu vermitteln, was in einem COMMODORE 16 steckt. Sie müssen es weiter versuchen mit neuen Programmen, auf die Ergebnisse achten und dabei mehr über Ihren Computer lernen, damit Sie Freude an Ihren Fortschritten haben. Und es gibt noch sooo viel zu programmieren.

Dieses Handbuch kann erste Schritte in der Computersprache BASIC vermitteln, stellt jedoch in keiner Weise ein Studier- oder BASIC-Lehrbuch dar. Jedoch gibt das nachfolgende BASIC-Lexikon eine Gesamtaufstellung ALLER BASIC-Befehle des COMMODORE 16, einschließlich Erklärungen und Beispielen. In vielen der vorangegangenen Programmbeispiele haben Sie vielleicht nicht jeden Schritt nachvollziehen können. Zu Ihrer Weiterbildung empfehlen wir Ihnen die Lektüre spezieller BASIC-Lehrbücher. Die COMMODORE-Sachbuchreihe bietet ein besonders interessantes Spektrum an weiterführender Literatur. In diesen Büchern ist das Hauptaugenmerk auf das 'WIE' und 'WARUM' der Programmierkunst gelegt, und es werden die ganz geheimen und die weniger geheimen Tricks verraten, wie das meiste aus Ihrem Computer herauszuholen ist.

Befürchten Sie nicht, nach der Lektüre dieses Handbuchs in Zukunft allein und ohne weitere Unterstützung zu sein. Es gibt eine Fülle einschlägiger Fachzeitschriften, USER-Gruppen und Vereine, die Ihnen gerne zur Seite stehen werden.

B A S I C 3.5

L E X I K O N

* EINLEITUNG
* KOMMANDO- UND ANWEISUNGSFORMAT
* KOMMANDOS BASIC 3.5
* ANWEISUNGEN BASIC 3.5
* WEITERE INFORMATIONEN ÜBER GRAFIK-ANWEISUNGEN
* FUNKTIONEN
* VARIABLEN UND OPERATOREN
* VARIABLEN-NAMEN
* MATRIZEN / FELDER
* RESERVIERTE VARIABLEN-NAMEN
* BASIC-OPERATOREN

```
****************
*  EINLEITUNG  *
****************
```

Die vielen Beispiele und Übungen dieses Handbuches haben Ihnen bereits einen guten Eindruck von der Computerprogrammierung in BASIC vermittelt. Das folgende Lexikon beschreibt den vollständigen Wortschatz und die Regeln (SYNTAX) der Programmiersprache BASIC 3.5. Jedes Sprachelement wird durch kurze Beispiele erläutert, die Sie durch eigenes Experimentieren fortführen und vertiefen sollten. Sie brauchen nicht zu befürchten, daß der COMMODORE 16 durch irgendeine falsche Eingabe beschädigt werden könnte. Nur ständige Übung führt zur Meisterschaft!

Im Lexikon finden Sie die Formatdefinitionen, zusammenfassende Erklärungen sowie Beispiele zu allen Sprachelementen von BASIC 3.5. Es kann jedoch keinen Lehrgang ersetzen; hier muß vielmehr auf einschlägige Lehrbücher verwiesen werden.

Bei den BASIC-Befehlen haben wir zwischen Kommandos und Anweisungen zu unterscheiden, wobei die Grenze zwischen beiden Bereichen nicht immer klar definiert ist. Kommandos werden hauptsächlich im DIREKT-Modus angewendet, während Anweisungen meist in ein Programm eingebaut sind. Sofern die Kommandos mit einer Zeilennummer versehen werden, sind sie (bis auf wenige Ausnahmen) auch im PROGRAMM-Modus einsetzbar. Umgekehrt lassen sich viele Anweisungen auch im DIREKT-Modus verwenden (also ohne vorangehende Zeilennummer). Innerhalb der getrennten Kapitel sind die Kommandos bzw. Anweisungen alphabetisch aufgeführt.

Das BASIC 3.5 LEXIKON ist folgendermaßen aufgebaut:

---

* KOMMANDOS: Befehle, die hauptsächlich dem Umgang mit Programmen dienen, z.B. für das Laden, Starten, Editieren, Abspeichern, Löschen und für die Diskettenverwaltung.

* ANWEISUNGEN: Befehle, die hauptsächlich innerhalb von numerierten Programmzeilen eingesetzt werden.

* FUNKTIONEN: String-, Numerische und Druck-Funktionen.

* VARIABLEN UND OPERATOREN: Die unterschiedlichen Variablen-Typen, zugelassene Variablen-Namen, arithmetische und logische Operatoren.

```
************************************
*  KOMMANDO- UND ANWEISUNGSFORMAT  *
************************************
```

Die in diesem Kapitel aufgeführten Befehle werden anhand von festen Formatregeln beschrieben, um sie so eindeutig wie möglich darzustellen. In den meisten Fällen sind noch eine Reihe von Beispielen mit den entsprechenden Befehlen angefügt. Das folgende Beispiel zeigt einige der Formatregeln, wie sie in Kommandos und Anweisungen verwendet werden:

```
BEISPIEL:  DLOAD"Programm-Name" [,D0,U8]
             !          !         ! !
             !          !         ! Zusätzlicher Parameter
             !          !         Zusätzlicher Parameter
             !          Parameter
             Schlüsselwort (KEYWORD)
```

Die Teile eines Befehls, die exakt nach Vorlage einzutippen sind, werden großgeschrieben und unterstrichen gedruckt. Eingaben, die von Fall zu Fall variieren, sind durch Hinweistexte normaler Groß-/Kleinschrift repräsentiert. Enthält die Formatmaske Trennzeichen (z.B. Komma oder Semikolon) oder Anführungszeichen (" "; meist bei Programm- oder Datei-Namen), dann sind sie fester Bestandteil des Formats und müssen mit eingegeben werden.

Allgemein ist zu beachten, daß alle Zeichen OHNE SHIFT-Taste einzugeben sind. Das gilt auch dann, wenn Sie den Computer in den Schreibmaschinen-Modus geschaltet haben und anstelle der in der Formatmaske stehenden Großbuchstaben kleine Buchstaben auf dem Bildschirm erscheinen. Geshiftete Buchstaben oder Graphikzeichen sind prinzipiell nur innerhalb von Anführungszeichen erlaubt! (einzige Ausnahme: Abkürzungen von Schlüsselwörtern).

SCHLÜSSELWÖRTER (auch reservierte Worte genannt) erscheinen in Großbuchstaben und unterstrichen. DIESE WÖRTER MÜSSEN SO EINGEGEBEN WERDEN, WIE SIE VORGEDRUCKT SIND, allerdings können viele dieser Schlüsselworte mit ihrer Abkürzung (siehe hierzu Anhang: ABKÜRZUNGEN) eingegeben werden.

Die Schlüsselwörter sind Hauptbestandteile der Programmiersprache BASIC, wie sie der COMMODORE 16 versteht. Sie sind der wichtigste Teil eines Befehls, weil sie dem Computer mitteilen, welche Aktion er als nächstes auszuführen hat. Diese Wörter sind reserviert und dürfen daher nicht als Bestandteil eines Variablennamens auftreten!

PARAMETER: Werden in Groß-/Kleinbuchstaben geschrieben. Parameter stellen den Teil eines Befehls dar, den Sie selbst festlegen müssen. Sie vervollständigen ein Schlüsselwort, indem Sie diesem wesentliche Zusatzinformationen liefern. Beispielsweise wird dem Computer per Schlüsselwort mitgeteilt, ein Programm zu laden, während Parameter 1 spezifiziert, welches Programm, und Parameter 2, aus welchem Disketten-Laufwerk (bei Verwendung eines Doppellaufwerks) das Programm zu laden ist. Zu den Parametern zählen Filenamen, Variable, Zeilennummern, usw. Das Zeichen # in einem Parameter steht immer für 'Nummer'.

ECKIGE KLAMMERN: Enthalten Zusatz-Parameter, deren Angabe nicht zwingend, sondern in gewissen Standard-Situationen entbehrlich ist.

SPITZE KLAMMERN: Enthalten eine Liste von Zusatz-Parametern, wovon einer zwingend gewählt werden muß.

AUSLASSUNGSZEICHEN (...): Eine Folge von drei Punkten bedeutet, daß eine Parameterliste um weitere, gleichartige Parameter verlängert werden kann (die jeweils durch Kommata getrennt werden).

TRENNSTRICH (|) : Werden die Teile einer Liste durch Trennstriche anstelle von Kommata getrennt, so handelt es sich um eine Auswahlliste alternativer Eingabemöglichkeiten.

VARIABLE: An dieser Stelle muß eine gültige BASIC-VARIABLE wie z.B. X, A$ oder T% eingetippt werden. An vielen Stellen ist jedoch auch eine ganze Formel (arithmetischer oder String-Ausdruck) zugelassen, die zur Sicherheit in Klammern eingeschlossen werden kann.

TERM: Dies ist jeder in BASIC gültige Formelterm wie z.B. A+B+2 oder .5*(X+3).

ANFÜHRUNGSZEICHEN, RUNDE KLAMMERN, KOMMA, SEMIKOLON, BINDESTRICH: Diese Zeichen sind fester Bestandteil eines Befehls und müssen stets eingegeben werden.

```
*****************************
*  KOMMANDOS IN BASIC 3.5   *
*****************************
```

AUTO
****
AUTO [Zeilenabstand]

Schaltet die automatische Zeilennumerierung ein, wodurch die Programmeingabe wesentlich erleichtert wird. Die nächste Zeilennummer wird nach jedem <Return> automatisch vorgegeben, und der CURSOR wartet in der richtigen Schreibposition. Der Klammerwert (Parameter) bestimmt, in welchem Zeilennummernabstand die nächste Zeile automatisch ausgegeben wird. Das Kommando AUTO ohne Parameter schaltet den Modus wieder aus; der RUN-Befehl bewirkt das gleiche. Der AUTO-Befehl kann nur im DIREKT-Modus eingegeben werden.

BEISPIELE:

AUTO 10	Numeriert Zeilen automatisch im Abstand von 10.
AUTO 50	Numeriert Zeilen automatisch im Abstand von 50.
AUTO	Schaltet automatische Zeilennumerierung AUS.

BACKUP
******
BACKUP DLaufwerk-# TO DLaufwerk-# [<,|ON> UGeräte-#]

Mit diesem Befehl werden bei einem Doppel-Laufwerk alle Files der einen Diskette auf die andere Diskette kopiert. Dabei kann die 'empfangende' Diskette noch unformatiert sein, da der BACKUP-Befehl alle Informationen der 'sendenden' Diskette - einschließlich Format - übernimmt. Wichtige Original-Disketten sollen stets mit diesem Befehl gesichert werden (Sicherung gegen Beschädigung oder Verlust!).

Das BACKUP-Kommando formatiert (headert) auch die Diskette, d.h. alle eventuell auf dieser Diskette vorhandenen Daten werden unwiederbringlich gelöscht - daher VORSICHT!. Siehe auch 'COPY'.

WICHTIG: Das BACKUP-Kommando kann nur bei einer Doppel-Floppy verwendet werden.

BEISPIELE:

BACKUP D0 TO D1	Kopiert die gesamte Diskette in Drive 0 auf die Diskette in Laufwerk 1 (Drive 1)
BACKUP D0 TO D1, U9 BACKUP D0 TO D1 ON U9 (BACKUP D0 TO D1,ON U9 ...	Kopiert die gesamte Diskette in Drive 0 auf die Diskette in Drive 1, Gerät 9 wird ebenfalls akzeptiert.)

COLLECT
****

COLLECT [DLaufwerk-#] [<,|ON> UGeräte-#]

Mit diesem Befehl wird die Diskette 'bereinigt', d.h. nicht korrekt geschlossene Files werden geschlossen und damit auch das Inhaltsverzeichnis auf den richtigen, freien Speicherplatz korrigiert.

BEISPIEL:

COLLECT D0

CONT (=CONTINUE)
****

Ein gestopptes Programm - egal ob per Taste <Run/Stop>, durch eine STOP- oder END-Anweisung in einem Programm - kann mit dem Befehl CONT wieder gestartet werden. Die Programmfortsetzung erfolgt genau hinter der Abbruchstelle.

Das Kommando CONT funktioniert nicht, wenn nach dem Stopvorgang Programmzeilen geändert, hinzugefügt oder entfernt wurden. Hielt das Programm wegen eines Programmfehlers (ERROR) an, oder Sie veranlaßten während des Stopvorgangs eine ERROR-Meldung, so ist ein Wiederstart (mit CONT) nicht möglich. In diesen Fällen erscheint die Fehlermeldung: CAN'T CONTINUE ERROR.

COPY
****
COPY [DLw-#,]"Quelle"TO[DLw-#,]"Ziel"[<,|ON> UGeräte-#]

Mit dem Befehl COPY kann ein bestimmtes File von einem Laufwerk (Quell-File) auf die Diskette im anderen Laufwerk (nur in einer Doppelfloppy möglich, Zieldiskette muß schon formatiert sein) unter dem gleichen Filenamen, oder im selben Laufwerk unter anderem Filenamen kopiert werden.

BEISPIELE:

COPY D0,"ABEND" TO D1,"NACHT"	Kopiert File "ABEND" von Laufwerk 0 auf Diskette in Drive 1 mit Umbenennung in "NACHT".
COPY D0,"ABEND" TO D1,"ABEND"	Kopiert File "ABEND" von Laufwerk 0 auf Diskette in Drive 1.
COPY D0 TO D1	Kopiert alle Files von Laufwerk 0 auf Diskette in Drive 1.
COPY "KATZEN" TO "HUNDE"	Kopiert File "KATZEN" auf demselben Laufwerk mit Umbenennung in File "HUNDE".
COPY D0,"DATEN*" TO D1,"*"	Kopiert von Laufwerk 0 alle Files, deren Namen mit "DATEN" anfangen, unter selbem Namen nach Laufwerk 1.
COPY D0,"?" TO D1,"*"	Kopiert von Laufwerk 0 alle Files, deren Namen genau ein Zeichen lang sind, genauso nach Laufwerk 1.

DELETE
******

DELETE [Erste Zeilen-#] [- [Letzte Zeilen-#]]

Dieses Kommando löscht BASIC-Programmzeilen (von - bis). Es ist nur im DIREKT-Modus anwendbar.

BEISPIELE:

DELETE 75	Löscht Programmzeile 75.
DELETE 10-50	Löscht die Zeilen 10 bis einschließlich Zeile 50.
DELETE -50	Löscht alle Zeilen vom Programmanfang bis einschließlich Zeile 50.
DELETE 75-	Löscht alle Zeilen des Programms ab Zeile 75 bis zum Programmende.

DIRECTORY
****

DIRECTORY [DLaufwerk-#] [<,|ON> UGeräte-#] [,"Filename"]

Um das Inhaltsverzeichnis einer Diskette auf den Bildschirm des COMMODORE 16 zu bringen, dient das Kommando DIRECTORY. Der Durchlauf (= Scrolling) des gelisteten Inhaltsverzeichnisses kann mit der Tastenkombination <Control> & <S> angehalten und mit jeder beliebigen Taste wieder gestartet werden. Mit der COMMODORE-Taste <C=> wird der Durchlauf verlangsamt. Mit dem Befehl DIRECTORY läßt sich keine Hardcopy (= Papierausdruck des Bildschirminhalts) erstellen. Um dies zu erreichen, muß das DIRECTORY in den Speicher geladen (eventuell vorhandene Programme werden dadurch überschrieben - und unbrauchbar, daher VORSICHT!) und wie ein Programm auf den Drucker gelistet werden.

BEISPIELE:

DIRECTORY	Listet alle Files der Diskette.
DIRECTORY D1, U9, "WALD"	Listet das File "WALD" der Diskette im Laufwerk 1 mit der Gerätenummer 9.

DIRECTORY D0, "PGM ?.VERS"	Das sog. Jokerzeichen '?' steht stellvertretend für alle Zeichen an dieser Textstelle: Die Files "PGM 1.VERS", "PGM 2.VERS", "PGM 3.VERS" passen dazu und werden daher gelistet.
DIRECTORY D1, "PGM.*"	Das Jokerzeichen "*" steht stellvertretend für alle möglichen Fortsetzungen: Dazu passen z.B. "PGM.", "PGM.ALT", "PGM.N-V1".

WICHTIG: Um das Inhaltsverzeichnis der Diskette im Laufwerk 0 des Geräts 8 auszudrucken, benutzen Sie die Kommandofolge:

```
LOAD "$0",8
OPEN 4,4: CMD 4: LIST
PRINT#4: CLOSE 4
```

DLOAD
*****

DLOAD "Filename" [,DLaufwerk-#] [<,|ON> UGeräte-#]

Mit diesem Befehl wird ein Programm bestimmten Namens in den vorhandenen Speicherbereich geladen. (Kassettenprogramme werden mit dem Befehl LOAD geladen). Im Gegensatz zum LOAD-Befehl von Kassette muß der Programmname angegeben werden, wobei die Jokerzeichen "?" und "*" verwendet werden dürfen.

BEISPIELE:

DLOAD "AUTOBUS"	Sucht auf der Diskette (im Inhaltsverzeichnis) den Filenamen "AUTOBUS" und lädt dieses Programm in den Speicher.
DLOAD (A$)	Lädt ein Programm, dessen Name in der Variablen 'A$' gespeichert ist. Ist A$ leer, kommt eine ERROR-Meldung.

Der DLOAD-Befehl kann auch aus einem BASIC-Programm heraus verwendet werden, um ein Programm zu finden und zu starten (RUN). Dieser Vorgang wird Verketten (= chaining) genannt. Das nachgeladene Programm muß kürzer als das vorhergehende sein, dann können die bisher verwendeten Variablen weiterverarbeitet werden.

DSAVE
*****

DSAVE "Filename" [,DLaufwerk-#] [<,|ON> UGeräte-#]

Programme werden mit diesem Befehl auf Diskette abgespeichert. (Bei Kassettenbetrieb lautet der Befehl SAVE). Es muß ein Programmname vergeben werden. Ein 'Klammeraffe' (bei DIN-Tastatur ein §) vor dem Namen bestimmt, daß ein vorhandenes Programm gleichen Namens zu überschreiben ist.

BEISPIELE:

DSAVE "§FREITAG"	Das Programm "FREITAG" auf Diskette überschreibend abspeichern.
DSAVE (A$)	Das Programm, dessen Name in der Variablen 'A$' abgelegt ist, auf Diskette speichern.
DSAVE "PGM 3",D0,U9	Speichert das Programm "PGM 3" auf die Diskette im Laufwerk 0 des Geräts 9.

HEADER
******

HEADER "Diskettenname", DLaufwerk-# [,IID] [<,|ON> UGeräte-#]

Bevor eine neue Diskette erstmals in einer Floppy eingesetzt werden kann, muß sie mit dem Befehl HEADER formatiert werden. Auch gebrauchte Disketten, die neu benützt werden sollen, lassen sich auf diese Weise 'erneuern' - mit dem Befehl HEADER. Der HEADER-Befehl veranlaßt, daß die Diskette in Spuren und Sektoren eingeteilt wird und daß dafür auch ein Verzeichnis (das DIRECTORY) auf der Diskette angelegt wird.

Als Diskettenname kommt jede Kombination bis zu einer Maximallänge von 16 Zeichen in Frage. Die Identität (= ID) besteht aus zwei Zeichen. Jede Diskette soll möglichst eine andere ID bekommen. Und Vorsicht mit dem HEADER-Befehl - er bewirkt, daß alle Daten auf der Diskette unwiederbringlich gelöscht werden!

Wenn Sie keine ID vergeben, erfolgt der HEADER-Vorgang wesentlich schneller - es bleibt dann bei der bisherigen ID. Verständlich, daß diese Methode nur bei gebrauchten Disketten funktionieren kann, da der schnelle HEADER-Vorgang lediglich das Inhaltsverzeichnis löscht und nicht die gesamte Diskette formatiert.

BEISPIELE:

HEADER "TEST DISK", I23, D0

HEADER "UEBUNGEN", I1A, D1, U8

HELP
****

HELP

Der HELP-Befehl wird nach Programmfehlern aufgerufen. Wenn Sie HELP eintippen (und mit Taste <Return> abschicken), dann wird die fehlerhafte Programmzeile gelistet, wobei der Teil mit dem Fehler blinkend dargestellt wird (s.a. HELP-Taste).

KEY
***

KEY [Funktionstasten-#, Zugeordneter String]

Der COMMODORE 16 verfügt über acht Funktionstasten: vier Tasten ohne und vier Tasten mit der Taste <Shift> zu bedienen. Unabhängig zu der durch den Einschaltvorgang vorgenommenen Standardbelegung, kann jede der acht Funktionstasten mit einem anderen String belegt werden.

Geben Sie KEY ohne Parameter ein, dann wird die momentane Belegung am Bildschirm gelistet. Der einer Funktionstaste zugeordnete String wird bei Betätigung dieser Taste am Bildschirm ausgedruckt. Die Gesamtlänge aller acht Strings zusammen darf 128 Zeichen nicht überschreiten. Jeder der acht Tasten können sowohl Einzel- wie auch Serienbefehle zugeordnet werden; wie zum Beispiel:

```
KEY 7, "GRAPHIC 0" + CHR$(13) + "LIST" + CHR$(13)
```

Dieses Belegungsbeispiel veranlaßt - sofern Funktionstaste 7 gedrückt und im DIREKT-Modus abgeschickt wird - die Umschaltung in den TEXT-Modus und ein anschließendes Programmlisting auf den Bildschirm. Mit dem String CHR$(13) wird ein RETURN-Befehl ausgelöst. Wenn Sie Anführungszeichen in dem String brauchen, dann schreiben Sie diese mit dem String CHR$(34).

Auch per Programm lassen sich die Funktionstasten umschreiben. Ein Beispiel:

```
10 KEY 2, "TEST VON" + CHR$(34): KEY 3, "NEIN"
```

oder

```
10 FOR I=1 TO 8: KEY I, CHR$(I+132): NEXT
```

Diese Programmzeile definiert die Funktionstasten, wie sie beim COMMODORE 64 und VC 20 belegt sind.

Um die Funktionstasten wieder mit ihren Standardwerten zu belegen, muß die Reset-Taste des COMMODORE 16 gedrückt (oder natürlich die geänderten Tasten durch KEY ... neu belegt) werden.

LIST
****

<u>LIST</u> [Erste Pgm-Zeile] [- [Letzte Pgm-Zeile]]

Mit dem LIST-Befehl kann ein BASIC-Programm Zeile für Zeile ausgedruckt werden, das vorher mit dem LOAD-Befehl in den Speicher des COMMODORE 16 geladen wurde. Wird der LIST-Befehl ohne zusätzliche Parameterangaben (ohne nachfolgende Zahlen) benützt, dann erscheint das gesamte Listing auf dem Bildschirm. Verlangsamt wird per COMMODORE-Taste <C=>, angehalten mit Taste <Control> & Taste <S>, wieder ausgelöst mit irgendeiner Taste und gestoppt mit der Taste <Run/Stop>.

Folgt dem LIST-Befehl nur eine Programmzeilennummer, so wird diese Zeile (sofern es sie im Speicher gibt) angezeigt. Werden mit dem LIST-Befehl zwei aufsteigende Zahlen, mit einem Bindestrich (Minussymbol '-') getrennt, geschrieben, dann wird das Programm von der ersten Zeilennummer bis einschließlich der zweiten Nummer gelistet. LIST mit einer Zahl und einem Bindestrich veranlaßt das Listing ab dieser Zeilennummer bis zum Programmende, und LIST mit Bindestrich und Zahl bewirkt ein Programmlisting vom Programmanfang bis zu dieser Zeilennummer. Beim richtigen Einsatz dieser Variationsmöglichkeiten läßt sich jeder Programmteil listen bzw. zur Modifikation auf den Bildschirm bringen.

BEISPIELE:

LIST	Zeigt das gesamte Programm.
LIST 100-	Zeigt von Zeile 100 bis zum Programmende alles.
LIST 10	Zeigt nur die Programmzeile 10.
LIST -100	Zeigt alles vom Programmanfang bis einschließlich Zeile 100.
LIST 10-200	Zeigt alle Programmzeilen ab Zeile 10 bis inclusive Programmzeile 200.

LOAD
****

LOAD "File-Name" [,Geräte-#] [,Speicheradresse-Flag]

Mit dem Befehl LOAD werden Programme, die entweder auf Kassette oder Diskette abgespeichert sind, in den Arbeitsspeicher des Computers geholt. Nach LOAD allein und der Taste <Return> erscheint die Aufforderung zum Drücken der <PLAY>-Taste. Ist das erfolgt, wird der Bildschirm des COMMODORE 16 blank, und die Datassette startet, um ein Programm auf der Kassette zu suchen. Das erste gefundene Programm wird mit FOUND "File-Name" gemeldet (der Rekorder hält an). Ein Druck auf die COMMODORE-Taste <C=> veranlaßt den Beginn des Ladevorgangs. Wird nichts getan, geschieht nach einer kurzen Pause automatisch dasselbe. Ein Abbruch ist mit der <Run/Stop>-Taste möglich. Ist ein Programm erst einmal im Speicher des COMMODORE 16, so kann es in der Folge mit RUN (und Taste <Return>) gestartet, mit LIST gelistet oder abgeändert werden.

Dem Befehl LOAD kann auch ein Programmname (stets zwischen Anführungszeichen " ") folgen, und nach dem Programmnamen ein Komma (',' - außerhalb der Anführungszeichen) und eine Zahl (oder Zahlenvariable). Diese Zahl gibt die Gerätenummer an, d.h. sie bestimmt, ob von Diskette oder Kassette geladen wird. Fehlt diese Zahl, so interpretiert der COMMODORE 16 dies als Gerätenummer 1 (= Datassette). Das Diskettenlaufwerk hat normalerweise die Gerätenummer 8.

BEISPIELE:

LOAD	Lädt das nächste auf Kassette befindliche Programm in den Speicher des COMMODORE 16.
LOAD "BASIS"	Sucht die Kassette nach dem Programm "BASIS" durch und lädt es, sofern gefunden.
LOAD A$	Sucht nach einem Kassettenprogramm, dessen Name in der Variablen A$ abgelegt ist.
LOAD "MAUERN",8	Es wird das Programm "MAUERN" auf Diskette gesucht und sofern vorhanden auch in den Speicher geladen.

Den LOAD-Befehl kann man auch in BASIC-Programme einbinden und auf diese Weise das nächste Programm von Kassette nachladen und starten. Dieser Vorgang wird als Verketten (Chaining) bezeichnet. Das zweite Programm muß in der Regel kürzer als das erste sein.

Das Speicheradresse-FLAG bestimmt, an welche Speicheradresse das Programm zu laden ist. FLAG = 0 informiert den COMMODORE 16, das Programm an den Anfang des BASIC-Speicherbereichs zu laden. FLAG = 1 lädt dagegen das Programm auf jeden Fall an die Adresse, die es beim Abspeichern innehatte. Standard (d.h. wenn keine Angabe erfolgt) ist FLAG = 0. Der Wert 1 wird im allgemeinen nur in Verbindung mit zu ladenden Maschinenprogrammen benützt.

NEW
***
<u>NEW</u>

Der BASIC-Befehl NEW löscht das gegenwärtige Programm im Speicher und setzt gleichzeitig alle verwendeten Variablen wieder auf Null. Wenn das Programm vorher nicht abgespeichert wurde, ist es bis zur Wiedereingabe verloren. Daher ist der Befehl NEW mit VORSICHT anzuwenden.

Auch der Befehl NEW kann in BASIC Programme eingebaut werden. Erreicht der COMMODORE 16 die Programmzeile mit dem NEW-Befehl, wird das gesamte Programm gelöscht und der Programmablauf abgebrochen. Das ist natürlich höchstens dann sinnvoll, wenn ein Programm nach vollständigem Ablauf gezielt beendet werden soll.

RENAME
******

RENAME "Alt" TO "Neu" [,DLaufwerk-#] [<,|ON> UGeräte-#]

Dieser RENAME Befehl wird eingesetzt, um einen Filenamen auf der Diskette umzubenennen.

BEISPIEL:

RENAME "VORTEIL" TO "SCHULDEN",D0	Tauscht auf der Diskette den Filenamen "VORTEIL" in den Namen "SCHULDEN" um.

RENUMBER
****

RENUMBER [Neue Startz-# [,Zeilenabstand [,Alte Startz-#]]]

Die neue Startzeilennummer bildet nach dem RENUMBER-Vorgang die erste Programmzeile. Erfolgt keine Angabe, so nimmt das System als Standard ('Default') Zeilennummer 10 an.

Zeilenabstand ist der numerische Abstand zwischen zwei Programmzeilen. Standard ist ebenfalls 10.

Die alte Startzeilennummer legt den Beginn des Renumber-Prozesses fest. Somit lassen sich auf diese Weise auch Teile des Programms neu durchnumerieren. Fehlt dieser Parameter, wird auf die erste Zeile Bezug genommen.

Der RENUMBER-Befehl kann nur im DIREKT-Modus eingesetzt werden.

BEISPIELE:

RENUMBER 20, 20, 1	Der RENUMBER-Vorgang startet bei der bisherigen Zeile 1, die zur Zeile 20 wird. Die Zeilennummern steigen ab da jeweils um 20.
RENUMBER , , 65	Beginnend ab Zeile 65, die nun Zeile 10 wird, steigen die Zeilennummern mit 10.

RUN
***
RUN [Zeilen-#]

Nachdem ein Programm in den Speicher des COMMODORE 16 eingetippt oder eingeladen worden ist, kann es durch den RUN-Befehl gestartet werden. RUN löscht vor Programmbeginn alle Variablen. Steht der Befehl RUN allein, so beginnt der Computer bei der niedrigsten Zeilennummer mit der Programmausführung. Wird der Befehl RUN mit einer Zahl verbunden, so stellt dieser Parameter die Zeilennummer dar, bei der der Programmablauf startet. Der Befehl RUN kann auch in ein BASIC-Programm eingebunden werden.

BEISPIELE:

RUN	Startet den Programmablauf bei der ersten Zeile.
RUN 100	Startet das Programm ab der Zeilenzahl 100.

SAVE
****
SAVE [" File Name" [,Geräte-# [,EOT-FLAG]]]

Mit dem Befehl SAVE wird ein gegenwärtig im Speicher befindliches Programm auf Kassette oder Diskette abgespeichert. Geben Sie nur SAVE und <Return> ein, ist der COMMODORE 16 auf Kassettenspeicherung programmiert. Für den Computer besteht keine Möglichkeit, festzustellen, ob sich an dieser Stelle der Kassette bereits ein Programm befindet. Daher empfehlen sich genaue Aufzeichnungen über die Bespielung der Kassetten.

Wird mit dem SAVE-Befehl auch der zwischen Anführungszeichen stehende Filename oder eine String-Variable eingegeben, dann speichert der COMMODORE 16 dieses Programm unter diesem Namen ab. In Zukunft ist dieses Programm dann leichter wiederzufinden und einzusetzen.

Soll auch die Geräte-Nummer eingegeben werden, dann ist diese nach den Anführungszeichen mit einem Komma und der Zahl oder einer Zahlenvariablen miteinzugeben. Gerät 1 ist die Datassette, und Nummer 8 aktiviert die Diskettenstation.

Nach der Zahl 1 für Kassette kann nach einem weiteren Komma noch eine Zahl folgen: das EOT-FLAG (End Of Tape = Ende des Bandes). Ist diese zweite Zahl eine Eins ('1'), so wird vom COMMODORE 16 am Ende des abgespeicherten Programms eine Markierung gesetzt, die im Falle eines bis zu dieser Marke erfolglosen Versuchs eines Einladevorgangs einen Abbruch mit der Meldung FILE NOT FOUND ERROR bewirkt.

Bei Speicherung auf Diskette kann die Laufwerksnummer mit nachfolgendem Doppelpunkt direkt vor den Filenamen geschrieben werden. Wird vor die Laufwerksnummer noch ein 'Klmmeraffe' (bei DIN-Tastatur Paragraph) geschrieben, wird ein schon bestehendes Programmfile überschrieben.

BEISPIELE:

SAVE	Speichert das Programm ohne Namen auf die Kassette ab.
SAVE "PAPIER"	Speichert das Programm unter dem Namen "PAPIER" auf Kassette ab.
SAVE A$	Speichert das Programm unter dem Variablen Namen aus A$ auf Kassette ab.
SAVE "EIGENTUM",8	Speichert das Programm unter dem Namen "EIGENTUM" auf Diskette ab.
SAVE "1:TAGEBUCH",8	Speichert das Programm unter dem Namen "TAGEBUCH" auf Laufwerk 1 der Floppy ab.
SAVE "§0:TAGEBUCH",8	Überschreibt das Programm "TAGEBUCH" auf Laufwerk 0 der Floppy.
SAVE "SPIELE",1,1	Speichert das Programm "SPIELE" mit dem EOT-FLAG auf Kassette ab.

SCRATCH
****

SCRATCH "File Name" [,DLaufwerk-#] [<,|ON> UGeräte-#]

Um ein File auf der Diskette zu löschen, bedient man sich des SCRATCH-Befehls. Als zusätzliche Sicherheit kommt nach der Eingabe die Frage: Are you sure? (=sind Sie sicher?). Wird diese Frage mit 'y' für 'yes' (=ja) beantwortet, beginnt erst der Löschvorgang. Mit 'n' für 'no' (=nein) wird der Vorgang abgebrochen. Der Befehl ist u.a. sehr nützlich, um wieder mehr Speicherplatz auf der Diskette zu schaffen, indem unerwünschte Files gelöscht werden.

BEISPIEL:

SCRATCH "VERSION 2", D1	Löscht das File "VERSION 2" auf der Diskette in Laufwerk 1.

VERIFY
******

VERIFY ["File Name" [,Geräte-# [,Speicheradressen-FLAG]]]

Der byteweise Vergleich eines auf Diskette gespeicherten Programms mit dem im Speicher des COMMODORE 16 befindlichen Programm erfolgt mit dem Befehl VERIFY. Soeben abgespeicherte Programme werden am besten sofort auf einwandfreie Speicherung damit überprüft.

Der Befehl VERIFY bewährt sich auch im Einsatz der Kassettenstation, weil damit exakt das Ende des gleichen (oder auch eines anderen - nur wird dies dann entsprechend gemeldet) Programms auf der Kassette gefunden werden kann. Das nächste Programm kann jetzt ab dieser Stelle ohne die Gefahr des Überschreibens abgespeichert werden.

Lautet der Befehl VERIFY ohne Parameterzusatz, dann wird auf Kassette einfach das nächste kommende Programm mit dem Programm im Speicher auf Gleichheit - ohne Beachtung des Programmnamens - geprüft. Wird der VERIFY-Befehl mit einem zwischen Anführungszeichen geschriebenen Programmnamen oder einem Variablennamen gestartet, dann wird ausschließlich dieses Programm gesucht und, wenn vorhanden, mit dem Programm im Speicher verglichen. Die erste Zahl nach dem Programmnamen bestimmt das Gerät des Datenträgers (1= Kassette, 8= Diskette). Das Speicheradressen-FLAG hat die gleiche Bedeutung wie im LOAD Befehl.

BEISPIEL:

VERIFY	Überprüft das nächste Programm auf Kassette auf Gleichheit mit dem im Speicher.
VERIFY "WIRKLICHKEIT"	Sucht das Programm "WIRKLICHKEIT" auf der Kassette und prüft es wenn gefunden gegen das im Speicher befindliche Programm.
VERIFY "EDEN",8,1	Sucht das Programm "EDEN" auf der Diskette im Gerät 8 (ab gespeicherter Adresse) und prüft es.

```
*******************************
*  ANWEISUNGEN IN BASIC 3.5   *
*******************************
```

BOX
***
<u>BOX</u> [Farbzonen-#], a1, b1, a2, b2, [,[Drehwinkel] [,Farbe]]

```
Farbzonen-# ..... Farbzone (0-3); Standard ist 1 (Vordergrund-Farbe)
a1, b1 .......... Eck-Koordinaten (skaliert - links, oben)
a2, b2 .......... Eck-Koordinaten (skaliert - rechts, unten); Standard
                  ist PC (=Pixel Cursor)
Drehwinkel ...... Drehung im Uhrzeigersinn (in Grad); Standard: 0 Grad
Farbe ........... Füllt Umrisse mit Farbe (0=AUS, 1=EIN); Standard: 0
```

Mit Hilfe der Anweisung BOX lassen sich Rechtecke jeder Größe auf den Bildschirm zeichnen. Um den Standardwert zu verwenden, genügt es, ein Komma zu schreiben - ohne Wert. Der Drehpunkt befindet sich in der Mitte des Rechtecks. Der Pixel-Cursor (PC) befindet sich (unsichtbar) in der Koordinate a2, b2, nachdem die BOX-Anweisung ausgeführt wurde.

BEISPIELE:

BOX 1, 10, 10, 60, 60	Zeichnet die Umrisse eines Rechtecks.
BOX , 10, 10, 60, 60, 45, 1	Zeichnet ein gedrehtes, eingefärbtes Rechteck (Rhombus).
BOX , 30, 90, , 45, 1	Zeichnet ein eingefärbtes, gedrehtes Vieleck (Polygon).

CHAR
****
CHAR [Farbzonen-#],x,y [,[String] [,Reverse-FLAG]]

```
Farbzonen-# ..... Farbzone (0-3)
x ............... Buchstaben/Zeichen-Spalte (0-39)
y ............... Buchstaben/Zeichen-Zeile  (0-24)
String .......... Zu druckender Text
Reverse ......... Reverse-Flag (0=AUS, 1=EIN)
```

Textzeilen (Alphanumerische Strings) können mit dem Befehl CHAR in jedem Grafik-Modus an der vorbestimmtem Position angezeigt werden. Die Buchstaben werden aus dem Zeichengenerator des COMMODORE 16 gelesen. Einzugeben sind lediglich die Anfangskoordinaten x und y sowie die darzustellende Textzeile. Die Farbzonennummer und das Reverse-FLAG sind keine zwingenden Zusatzinformationen.

Läuft eine Textzeile über den rechten Rand hinaus, dann wird sie in der nächsten Zeile fortgesetzt. Ist der TEXT-Modus eingeschaltet, dann verhält sich der mit der CHAR-Anweisung gedruckte String wie ein mit dem PRINT-Befehl gedruckter String, einschließlich Revers-, CURSOR-, Blinken- (Ein/Aus) Darstellungen, etc. Diese Kontrollfunktionen innerhalb des Strings funktionieren nicht, wenn für die Darstellung von Text im Grafik-Modus die Anweisung CHAR verwendet wird.

WICHTIG: Wenn im MEHRFARBEN-Modus ein Buchstabe/Zeichen im Mehrfarbenbereich 2 dargestellt werden soll, so ist die Farbzonen-# auf 0 und das Reverse-FLAG auf 1 zu setzen. Darstellung im Mehrfarbenbereich 1 erfordert Farbzonen-# und Reverse-FLAG auf 0.

CIRCLE
******

CIRCLE [cs], [a,b], xr, [yr], [sa], [ea], [Winkel], [inc]

```
cs ............. Farbzone (0-3)
a, b ........... Mittelpunkt-Koordinaten (skaliert)
                 Standard ist die Position des PC (=Pixel-Cursor)
xr ............. Radius in X-Achse (skaliert)
yr ............. Radius in Y-Achse (als Standard wird xr übernommen)
sa ............. Winkel (in Grad) am Kreisbogen-Anfang (Standard = 0)
ea ............. Winkel (in Grad) am Kreisbogen-Ende (Standard = 360)
Winkel ......... Drehung im Uhrzeigersinn (in Grad); Standard = 0 Gd.
inc ............ Winkel zwischen zwei Segmenten; Standard = 2 Grad
```

Mit der Anweisung CIRCLE lassen sich Kreise, Ellipsen, Kreissegmente, Dreiecke oder auch regelmäßige Vielecke zeichnen. Nach der Ausführung befindet sich der Pixel-Cursor auf dem Kreisumfang am Ende des Kreisbogens. Jede Drehung erfolgt um den Mittelpunkt. Kreisbögen werden durch regelmäßige Vielecke approximiert und vom Anfangspunkt im Uhrzeigersinn zum Endpunkt gezeichnet. Mit dem Winkel zwischen zwei Segmenten steuern Sie die Glätte der Rundung; je kleiner der Winkel, desto runder wird der Umfangsbogen. (Formel für die Anzahl der Ecken im Vollkreis: 360/inc).

BEISPIELE:

```
CIRCLE , 160,100,65,10            Zeichnet eine Ellipse
CIRCLE , 160,100,65,50            Zeichnet einen Kreis
CIRCLE , 60,40,20,18,,,,45        Zeichnet ein Achteck
CIRCLE , 260,40,20,,,,,90         Zeichnet eine Raute
CIRCLE , 60,140,20,18,,,,120      Zeichnet ein Dreieck
```

CLOSE
*****
CLOSE [File-#]

Mit der Anweisung CLOSE wird jedes geöffnete File abgeschlossen. Ein File wird gezielt über die der Anweisung folgende Zahl (= File-#) angesprochen.

BEISPIEL:

CLOSE 2 Das logische File 2 wird geschlossen.

CLR
***
CLR

Mit der Anweisung CLR werden zwar alle Variablen auf Null gesetzt, das im Speicher befindliche Programm bleibt hingegen erhalten. Die CLR-Anweisung ist automatisch im RUN- oder NEW-Befehl enthalten; aber auch mit jedem EDITIER-Vorgang (z.B. Programmzeilen ändern oder ergänzen) wird automatisch der CLR-Befehl ausgeführt.

CMD
***
CMD File-#

Ausgaben, die normalerweise zum Bildschirm gesendet werden (wie z.B. per PRINT-Anweisungen oder LIST-Befehl, jedoch keine POKE-Befehle auf den Bildschirm), werden mit dem Befehl CMD zu einem anderen Gerät umgeleitet. Dies können ein Drucker oder Daten-Files auf Kassette oder Diskette sein. Dieses Gerät bzw. File muß vorher jedoch geöffnet werden. Dann folgt der CMD-Befehl mit der Zahl oder numerischen Variablen, die die File-# repräsentiert. Wichtig: Vor dem CLOSE-Befehl auf das angegebene File immer einen PRINT#-Befehl mit gleicher File-# geben, sonst kann es zu merkwürdigen Bildschirmreaktionen kommen.

BEISPIELE:

OPEN 1,4	Öffnet Gerät #4, also den Drucker.
CMD 1	Die gesamte Ausgabe geht zum Drucker.
LIST	Das Listing erfolgt jetzt nicht am Bildschirm, sondern am Drucker - einschließlich READY.
PRINT #1	Schaltet die Ausgaben zurück zum Bildschirm.
CLOSE 1	Schließt das logische File 1.

COLOR
*****

<u>COLOR</u> Farbzonen-#, Farb-#, [, Helligkeitswert]

Mit der COLOR-Anweisung wird eine der fünf Farbzonen bestimmt:

Farbzonen-#	Bezeichnung
0	Bildschirm-Hintergrund
1	Vordergrund (Buchstaben, Zeichen)
2	Mehrfarben 1
3	Mehrfarben 2
4	Bildschirmrand

Die zweite Zahl in der COLOR-Anweisung selektiert die Hintergrundfarbe des gewählten Bildschirmbereichs. Diese Zahl ist mit den Farbtasten der Tastatur identisch.

Farb-Nr.	Farbe	Farb-Nr.	Farbe
1	Schwarz	9	Orange
2	Weiß	10	Braun
3	Rot	11	Gelb/Grün
4	Zyan	12	Rosa
5	Purpur	13	Blau/Grün
6	Grün	14	Hellblau
7	Blau	15	Dunkelblau
8	Gelb	16	Hellgrün

Jede Farbe ist auch noch in ihrer Helligkeit (LUMINANZ) veränderbar. Im Anhang an die Farbnummer kann diese LUMINANZ von Null ('0'= dunkel) bis sieben ('7'= hell) variieren. Der Standardwert für die Helligkeit ist 7. Die acht Helligkeitsstufen gelten für alle Farben, mit Ausnahme von Schwarz.

DATA
****
<u>DATA</u> Liste von Elementen, die durch Kommata getrennt sind

Der DATA-Anweisung folgt eine Liste von Elementen, die mit der READ-Anweisung vom Programm gelesen werden. Diese Posten können Zahlen oder Strings sein und müssen durch Kommata getrennt sein. Strings benötigen keine Anführungszeichen, außer es gehören dazu: Leerstellen, Doppelpunkt oder Komma. Steht zwischen zwei Kommata kein Element, so liest der Computer entweder eine Null oder einen Leerstring, je nach Variablentyp (s. READ), dafür ein. Siehe auch die RESTORE-Anweisung; mit ihr setzt der COMMODORE 16 den DATA-Zeiger auf das erste Element der (angegebenen) DATA-Zeile.

BEISPIEL:

```
DATA 100, 200, FRED, "WILMA:", , 3, 14, ABC123
```

DEF FN (=DEFiniere FuNktion)
******
<u>DEF FN</u> Name der Variablen = Formel

Mit der Anweisung DEF FN läßt sich eine komplexe Funktion mit einem kurzen Namen (Variable) definieren. Handelt es sich um eine komplexe und lange Formel, die oft aufgerufen wird, so kann mit Hilfe der DEF FN Anweisung in einem Programm viel Speicherplatz gespart werden.

Der Name, den Sie der Funktion geben, beginnt stets mit FN, gefolgt von irgend einem Namen. Bei der Definition muß die Funktion mit der Anweisung DEF, gefolgt vom zugeteilen Namen, angegeben werden. Dann folgt die in Klammern eingeschlossene numerische Variable (in unserem Beispiel 'X'). Nach dem Gleichheitszeichen folgt die zugewiesene Formel. Diese Formel kann jederzeit aufgerufen werden, wenn die Variable X durch eine Zahl ersetzt wird (siehe Beispiel Zeile 20):

BEISPIEL:

10 DEF FN A(X) = 12*(34.75-X/.3)+X

Mit der nachstehenden Programmzeile wird für die Variable X an allen Stellen der Wert 7 eingesetzt.

20 PRINT FN A(7)

Nach RUN und <Return> wird die Zahl 144 ausgegeben.

DIM
***
<u>DIM</u> Variable (Anzahl [, Anzahl ...]) [, ...]

Bevor ein Feld mit Variablen benutzt werden kann, muß das Programm die DIM-Anweisung durchlaufen, um den gewünschten Bereich einzurichten. Bei eindimensionalen Feldern mit weniger als 11 Elementen kann jedoch auf diese Anweisung verzichtet werden.

Der DIM-Anweisung folgt der Name (jeder zulässige Variablenname) für die Matrix. Danach in Klammern die Anzahl (auch Variablen zulässig) der einzurichtenden Elemente je Dimension. Von Matrix spricht man, wenn ein Feld mit mehreren Dimensionen errichtet wird. Ein Feld ist in seiner Dimension nur durch den benötigten Speicherplatz begrenzt. Um den Gesamtspeicherbedarf eines Feldes zu errechnen, multiplizieren Sie die Elementanzahlen jeder Dimension (plus 1, da die erste Nummer immer 0 ist) mit dem Speicherbedarf eines einzelnen Elements (Real 5 Bytes, Integer 2, Strings 3-258).

WICHTIG: Ganzzahlige (Integer-) Felder nehmen nur 2/5 des Speichers in Anspruch, den Fließkomma-Felder benötigen.

BEISPIEL:

```
10 DIM A$(40), B7(15), CC%(4,4,4)
       !       !         !
       !       !         125 (= 5*5*5) Elemente, 250 Bytes
       !       16 Elemente, 80 Bytes
       41 Elemente
```

Mit einer DIM-Anweisung lassen sich auch mehrere Felder dimensionieren, indem die Felder mit Komma getrennt werden. Wird die DIM-Anweisung vom Programm her mehr als einmal durchlaufen, wird das Programm mit der Fehlermeldung REDIM'D ARRAY ERROR abgebrochen. Plazieren Sie daher die DIM-Anweisung stets am Programmanfang.

DO/LOOP/WHILE/UNTIL/EXIT

****

DO [UNTIL Boolesches Arg./WHILE Boolesches Arg.]
[Anweisung(en) EXIT]
LOOP [UNTIL Boolesches Arg./ WHILE Boolesches Arg.]
(Ein Beispiel eines Booleschen Arguments ist A=1 oder H=>57)

Damit werden die Anweisungen zwischen der DO-Anweisung und der LOOP-Anweisung durchgeführt. Werden weder die DO- noch die LOOP-Anweisung von einem nachfolgenden WHILE- oder UNTIL-Parameter abgeschlossen, so erfolgt die Ausführung der dazwischenliegenden Anweisungen unbegrenzt oft. Ist in die DO-Schleife eine EXIT-Anweisung eingebaut, so geht die weitere Programmdurchführung auf die erste der LOOP-Anweisung folgende Anweisung über. DO-Schleifen können nach den FOR-NEXT-Schleifenregeln verschachtelt werden.

Wird UNTIL verwendet, wird die Schleife solange ausgeführt, bis (nach Booleschen Regeln) die Anweisung 'wahr' wird. WHILE stellt praktisch das Gegenteil von UNTIL dar: das Programm durchläuft die Schleife, solange die Anweisung 'wahr' ist.

BEISPIEL:

```
DO UNTIL X=0 OR X=1
    :
        LOOP
DO WHILE A$="": GET A$: LOOP
```

DRAW
****
DRAW [Farbzonen-#], [a1, b1,] [TO a2, b2] , ...

Mit dieser Anweisung lassen sich einzelne Punkte, Linien und Formen zeichnen. Es können die vier Farbzonen-Nummern ebenso angesprochen werden, wie die Anfangs- (a1, b1) und Endpunkte (a2, b2).

BEISPIELE:

Ein PUNKT:	DRAW 1, 100, 50	Kein Endpunkt spezifiziert: bewirkt, daß die Werte a1, b1 für a2, b2 gelten und dadurch ein Punkt entsteht.
Eine LINIE:	DRAW , 10,10, TO 100,60	
	DRAW , TO 25,30	
Ein DREIECK:	DRAW , 10,10 TO 10,60 TO 100,60 TO 10,10	

END
***
END

Erreicht ein laufendes Programm in einer Programmzeile eine END-Anweisung, so stoppt das Programm unverzüglich. Mit dem CONT-Befehl läuft das Programm ab der Anweisung weiter, die der END-Anweisung folgt.

FOR...TO...STEP
****
<u>FOR</u> Variable <u>=</u> Startwert <u>TO</u> Endwert [<u>STEP</u> Schrittweite]

Zusammen mit der NEXT-Anweisung veranlaßt die FOR-TO-Anweisung, daß ein bestimmter Programmabschnitt mehrfach durchlaufen wird. Auf diese Weise lassen sich z.B. Programmpausen unterschiedlichster Länge programmieren, Zählvorgänge durchführen oder bestimmte Arbeiten (z.B. Druckvorgänge) n-fach ausführen.

Nach jedem FOR-NEXT-Durchlauf wird die Schleifenvariable um den Wert 'Schrittweite' hinauf- oder heruntergezählt, wobei der Start- und der Endwert die Grenzwerte der Schleifenvariablen bilden.

Die FOR-Anweisung verläuft nach folgender Logik:
Zuerst wird die Schleifenvariable auf den Startwert gesetzt. Erreicht das Programm eine NEXT-Anweisung, wird die Schrittweite (Standard = 1) zum momentanen Wert der Schleifenvariablen hinzugezählt und danach das Ergebnis mit dem Endwert verglichen.

Ist das Ergebnis kleiner oder gleich (also nicht größer), wird der unmittelbar der FOR-Anweisung folgende Programmteil ausgeführt. Ist ein über dem Endwert liegender Wert erreicht, kommt der Programmteil zur Ausführung, der direkt auf die NEXT-Anweisung folgt. Siehe auch die NEXT-Anweisung. Bei negativer Schrittweite gilt sinngemäß das gleiche.

BEISPIEL:

```
10 FOR L = 1 TO 20
20 PRINT L
30 NEXT L
40 PRINT "BLACK JACK! L = " L
```

Dieses kleine Programm schreibt die Zahlen von Eins bis Zwanzig auf den Bildschirm und schließt mit der Meldung: BLACK JACK! L = 21.

Der FOR-TO-Anweisung kann die Anweisung STEP und die Schrittweite in Form eines arithmetischen Ausdrucks folgen. In diesem Fall wird der der Anweisung STEP folgender Wert statt der Standard-Schrittweite von Eins bei jedem Durchlauf zur Schleifenvariable hinzugezählt.

Schleifen können verschachtelt werden. Verschachtelte Schleifen müssen sorgfältig angelegt werden, und zwar so, daß die äußere Schleife erst dann weiterzählt, wenn die innere ganz durchlaufen ist.

BEISPIEL VERSCHACHTELTER SCHLEIFEN:

```
10 FOR L = 1 TO 100
20 FOR A = 5 TO 11 STEP 2      Diese FOR ... NEXT Schleife ist inner-
                               halb der äußeren (größeren)
30 NEXT A                      verschachtelt.
40 NEXT L
```

GET
***

<u>GET</u> Variable (meist String-Variable) [, ...]

Mit der GET-Anweisung ist es möglich, einzelne Zeichen über die Tastatur in den Computer zu bekommen. Erreicht ein Programm während der Ausführung eine GET-Anweisung, wird die Tastatur abgefragt. Wurde keine Taste gedrückt, so nimmt die GET-Anweisung dies als Null-Zeichen (Leer-String) an. Das Programm läuft weiter, ohne auf eine Tasteneingabe zu warten. Es besteht bei dieser (Eingabe-) Anweisung keine Notwendigkeit, die Taste <Return> zu drücken - vielmehr ist es mit der GET-Anweisung sogar möglich, die Taste <Return> als eigene Eingabe auszuwerten.

Der GET-Anweisung folgt stets ein Variablenname, meist eine Stringvariable. Wird eine numerische Variable benützt und die Eingabe ist keine Zahl, dann stoppt das Programm mit einer Fehlermeldung. Auch kann die GET-Anweisung in einer Schleife eingesetzt werden, wobei ständig nach einer Leereingabe (d.h. es erfolgt kein Tastendruck) abgefragt wird. Solange keine Tastatureingabe erfolgt, verweilt das Programm in dieser Schleife. Für diese Eingabetechnik bietet sich auch die GETKEY-Anweisung an. GET und GETKEY können nur im Programm-Modus angewendet werden !

BEISPIEL:

```
10 GET A$: IF A$ <> "A" THEN 10
```

Diese-GET Schleife wird solange durchlaufen, bis die Taste <A> gedrückt wird - dann erst läuft das Programm weiter.

GETKEY
******

<u>GETKEY</u> Variable (meist String-Variable) [, ...]

Die GETKEY-Anweisung ist der GET-Anweisung sehr ähnlich. Im Gegensatz zur GET-Anweisung wird bei der GETKEY-Anweisung keine Schleife benötigt, um das Programm an dieser Eingabestelle anzuhalten. GETKEY wartet mit dem Programmablauf, bis eine Taste gedrückt wird. Damit lassen sich auf einfache Weise Einzelabfragen der Tastatur programmieren. Diese Anweisung ist nur in einem Programm anwendbar.

BEISPIEL:

```
10 GETKEY A$
```

Bei dieser Programmzeile wartet das Programm, bis eine Taste gedrückt wird. Mit jedem Tastendruck läuft das Programm weiter.

GET#
****
GET# File-Nummer, Variable [, ...]

Die GET#-Anweisung wird benutzt, um über einen zuvor mit OPEN geöffneten Kanal ein Zeichen von einem bestimmten Gerät einzulesen. Es wird auf das Zeichen gewartet, ein Leerstring ist in Wirklichkeit CHR$(0). Ansonsten funktioniert die GET#-Anweisung wie die GET-Anweisung, ist also auch nur im Programm-Modus einsetzbar.

BEISPIEL:

```
100 GET#1, A$
```

GOSUB
*****
GOSUB Zeilen-#

Im Gegensatz zur GOTO-Anweisung merkt sich der COMMODORE 16 die Stelle, an der er die GOSUB-Anweisung erhielt. Erreicht das Programm in der Folge eine Programmzeile mit der Anweisung RETURN, springt das Programm automatisch zurück auf den Befehl, der der GOSUB-Anweisung folgt. Der von der GOSUB-Anweisung abgedeckte Bereich wird Subroutine genannt.

Eine Subroutine ist u.a. dann nützlich, wenn sie von verschiedenen Teilen des Programms aus angesprungen wird. Anstatt einen gleichen Programmteil an den unterschiedlichen Stellen immer wieder zu programmieren, erstellen Sie eine Subroutine und springen sie mit der GOSUB Anweisung von den verschiedenen Stellen aus an. Siehe auch die RETURN Anweisung.

BEISPIEL:

```
20 GOSUB 800
    :
    :
800 PRINT "HIER SIND WIR": RETURN
```

bedeutet, daß das Programm zu der bei Zeile 800 beginnenden Subroutine springen muß und dort weiter arbeitet ... bis zum RETURN.

GOTO oder GO TO
**** *****
GOTO Zeilen-#

Nach der Ausführung einer GOTO-Anweisung fährt das Programm am Anfang der angegebenen Zeile fort. Im DIREKT Modus angewandt, ermöglicht die Anweisung GOTO Zeilen-# einen Programmstart ab der eingegebenen Zeilenzahl - ohne dadurch auch sämtliche Variable (wie dies bei RUN der Fall ist) zu löschen.

BEISPIEL:

```
10 PRINT "UEBUNG MACHT DEN MEISTER"
20 GOTO 10
```

Die GOTO-Anweisung in Zeile 20 bewirkt einen ständigen Programmlauf - bis die Taste <Run/Stop> gedrückt wird.

GRAPHIC
****

GRAPHIC Modus [, Clear-Parameter]
GRAPHIC CLR

Der COMMODORE 16 kann in einen der fünf nachstehenden Grafik-Modi versetzt werden:

Modus	Grafik-Modus
0	Text
1	Hochauflösende Grafik
2	Hochauflösende Grafik und Text
3	Mehrfarben-Grafik
4	Mehrfarben-Grafik und Text

Clear-Parameter	Modus
0	Bildschirm wird n i c h t gelöscht
1	Bildschirm wird gelöscht

Bei der Ausführung einer der GRAPHIC-Anweisungen 1 bis 4 wird ein 10 KByte 'bit-mapped' Bereich (= HI-RES-Bereich) abgetrennt, und der Anfang des BASIC-Speicherbereichs über diesen Bereich verlegt. Dieser Hi-Res-Bereich bleibt auch gesperrt, wenn mit der Anweisung GRAPHIC 0 in den Text-Modus zurückgeschaltet wird. Enthält die GRAPHIC-Anweisung im zweiten Parameter eine Eins (z.B. GRAPHIC 0,1), dann wird mit der Umschaltung gleichzeitig der Bildschirm gelöscht.

Mit der Anweisung GRAPHIC CLR wird jedoch der Hi-Res Bereich-(10 KByte = 10240 Byte) wieder als BASIC-Speicherbereich für BASIC-Text und Variable freigegeben.

BEISPIELE:

GRAPHIC 3,1	Wählt den Hi-Res-Grafik-Modus und löscht den Bildschirm.
GRAPHIC CLR	Löscht den GRAPHIC-Bereich und gibt den gesamten Speicherbereich wieder für BASIC frei.

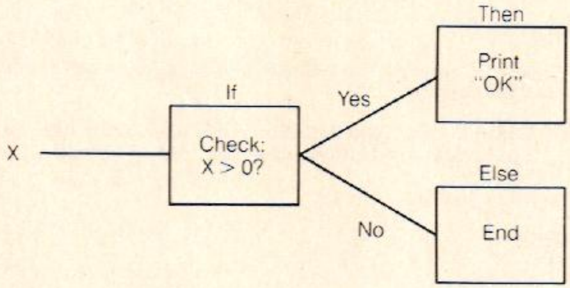
IF...THEN [..:ELSE]
**   ****   ****

<u>IF</u> Bedingung <u>THEN</u> wenn/dann-Klausel [:<u>ELSE</u> sonst-Klausel]

Mit der IF...THEN-Anweisung ist der COMMODORE 16 in der Lage, abhängig von einer bestimmten Situation zu entscheiden, welche Aktionen auszuführen sind. Wird die Bedingung erfüllt (d.h. ist sie im Booleschen Sinn 'wahr'), dann wird das Programm entsprechend der THEN-Anweisung weiter ausgeführt. Ist die Bedingung nicht erfüllt ('falsch'), dann wird das Programm in der nächsten Zeile fortgesetzt - es sei denn, daß auch noch ein ELSE-Teil besteht.

Die zu erfüllende Bedingung kann eine Variable oder eine Formel sein, wobei der Zustand 'wahr' eintritt, wenn die Bedingung einen Wert ungleich Null ergibt, und der Zustand 'falsch', wenn die Abfrage Null ergibt. In den meisten Fällen enthält diese Bedingung jedoch logische Operatoren (wie =, <, >, <=, =>, <>, AND, OR, NOT).

Wenn ein ELSE-Teil existiert, muß er in derselben Zeile wie die IF...THEN-Anweisung stehen. Der nach ELSE stehende Befehl wird dann ausgeführt, wenn die Bedingung nicht zutraf.



BEISPIEL:

```
50 IF X>0 THEN PRINT "OK": ELSE PRINT "FEHLER"
```

Mit dieser Anweisung wird der Wert X überprüft. Ist X>0, dann wird der THEN-Teil und nicht der ELSE-Teil ausgeführt. Ist X<=0, dann wird der ELSE-Teil ausgeführt und nicht der THEN-Teil.

INPUT
*****

INPUT ["Kommentar";] Variable [, ...]

Mit der INPUT-Anweisung können in einem Programm eine oder mehrere Variablenwerte an den COMMODORE 16 übergeben werden. Dazu hält das Programm an, gibt ein Fragezeichen ('?') auf den Bildschirm aus und wartet, bis eine Eingabe erfolgt, die mit der Taste <Return> abgeschlossen werden muß. Danach läuft das Programm wieder weiter.

Dem Wort INPUT folgt der Variablenname oder eine durch Kommata getrennte Variablenliste. Vor der Variablen kann ein Kommentar zwischen Anführungszeichen (" ") stehen, der sich auf die INPUT-Anweisung bezieht. Wird mit einem Kommentar gearbeitet, so ist dieser nach dem zweiten Anführungszeichen von der Variablen mit einem Semikolon (';') zu trennen.

Wenn Sie mehrere Werte einlesen wollen, müssen sie bei der Eingabe durch Kommata getrennt werden. Diese Anweisung ist nur im Programm-Modus zulässig!

BEISPIEL :

```
10 INPUT "Werte=";A,B,C
RUN
Werte =?  3,4,5.5
```

Werden weniger Werte eingegeben, als Variable vorhanden sind, so werden die fehlenden Werte durch zwei Fragezeichen nachgefordert.

BEISPIEL:

```
10 INPUT "WIE HEISSEN SIE";A$
20 INPUT "UND IHRE LIEBLINGSFARBE";B$
30 INPUT "WIE SCHNELL FLIEGT EINE SCHWALBE";A
```

INPUT#
******

<u>INPUT#</u> File-Nummer, Variable [, ...]

Die Anweisung INPUT# funktioniert wie die INPUT-Anweisung, nur daß die Daten von einem vorher geöffneten (OPEN) File oder Gerät kommen. Ein Kommentar ist in diesem Fall nicht gestattet. Auch diese Anweisung ist nur innerhalb eines Programms zulässig.

BEISPIEL:

```
50 INPUT#2, A$, C, D$
```

Hierbei müssen die Werte auch im File durch Kommata getrennt sein!

LET
***

<u>LET</u> Variable <u>=</u> Formel

Die LET-Anweisung wird benutzt, um einer Variablen einen Wert zuzuweisen. Das Wort LET ist jedoch optional, d.h., es kann auch weggelassen werden. Bei jeder Zuweisung ist sie automatisch (auch ungeschrieben) implementiert. Dabei befindet sich die empfangende Variable (die z.B. das Ergebis einer Berechnung aufnehmen soll) stets links vom Gleichheitszeichen ('=') und der zuzuweisende Wert (die Zahl oder die Formel) immer rechts davon.

BEISPIEL:

```
10 LET A = 5
20 B = 6
30 C = A < B+3
40 D$ = "HALLO"
```

LOCATE
******
LOCATE x-Koordinate, y-Koordinate

Mit der Anweisung LOCATE kann der PC (= Pixel-Cursor) an jede Stelle des Bildschirms plaziert werden. Der PC stellt die momentane Position des für die nächste Zeichnung notwendigen Startpunktes dar. Im Gegensatz zum regulären CURSOR ist der PC nicht als blinkender Punkt sichtbar - aber er ist mit der LOCATE-Anweisung dennoch bewegbar. Zum Beispiel:

LOCATE 160, 100

bringt den PC in die Mitte des Hi-Res-Bildschirmteils. Solange Sie nicht mit der Zeichnung beginnen, werden Sie nichts sehen. Mit Hilfe der RDOT (0)-Funktion können Sie jederzeit herausfinden, auf welcher x-Koordinate, und mit RDOT (1), auf welcher y-Koordinate sich der PC befindet. Mit der Anweisung PRINT RDOT (2) erfahren Sie auch, welcher Farbzonen-Nummer sich der PC gerade bedient. (Sie erinnern sich, daß in allen Zeichen-Anweisungen mit Farbwahl ein entsprechender Farbzonenwert zwischen 0 und 3 auszuwählen ist - entsprechend Vordergrund, Hintergrund, Mehrfarben 1, Mehrfarben 2).

MONITOR
****
MONITOR

Um in den eingebauten Maschinensprachenmonitor zu gelangen, bedienen Sie sich der Anweisung MONITOR. Der Monitor dient der Entwicklung, Fehlerbehebung (Debugging) und Ausführung von in Maschinensprache geschriebenen Programmen. Es empfiehlt sich das Studium weiterführender Literatur.

Um zurück in den BASIC-Modus zu gelangen, tippen Sie ein 'X' ein und drücken die Taste <Return>. Siehe auch im Anhang.

NEXT
****
NEXT [Variable,...,Variable]

Die NEXT-Anweisung wird in Verbindung mit der FOR-Anweisung gebraucht. Erreicht der Computer im Programmablauf eine NEXT-Anweisung, springt er zur zugehörigen FOR-Anweisung zurück und prüft die Schleifen-Variable. (Für mehr Detailinformationen siehe Anweisung FOR). Ist der Schleifendurchlauf beendet, wird das Programm mit der nächsten Programmanweisung hinter der NEXT-Anweisung fortgesetzt. Auf das Wort NEXT folgen meist eine oder mehrere Variablen, die durch Kommata getrennt sind.

Folgt kein Variablenname der Anweisung NEXT, so wird an dieser Stelle die zuletzt im Programm begonnene Schleife geschlosssen. Sind jedoch mehrere Variablen angegeben, so müssen sie entsprechend der Verschachtelung aufgeführt werden.

BEISPIEL:

```
10 FOR L = 1 TO 10: NEXT
20 FOR L = 1 TO 10: NEXT L
30 FOR L = 1 TO 10: FOR M = 1 TO 10: NEXT M, L
```

ON
**

ON Formel <GOTO/GOSUB> Zeilen-# 1 [, Zeilen-# 2,...]

Mit der ON-Anweisung lassen sich GOTO- oder GOSUB-Anweisungen in eine spezielle Version der IF-Anweisung verwandeln. Dem Wort ON folgt zuerst eine Formel, dann entweder die GOTO- oder die GOSUB-Anweisung und dann die durch Kommata getrennten Zeilennummern. Ergibt das numerische Ergebnis der Formel eine Eins, dann wird die erste der Zeilennummern angesprungen. Ist das Resultat eine Zwei, dann die zweite Zeilennummer, usw.

Ist das Ergebnis Null oder größer als die Anzahl der angegebenen Zeilennummern, dann wird die nächste auf die ON-Anweisung folgende Anweisung ausgeführt. Ergibt sich eine negative Zahl, stoppt das Programm mit der Meldung ILLEGAL QUANTITY ERROR.

BEISPIEL:

```
10 INPUT X: IF X<0 THEN 10  Wenn X=1, dann verzweigt die Anweisung ON
                            zur ersten Zeilennummer der Liste (50).
20 ON X GOTO 50, 30, 30, 70 Wenn X=2, dann verzweigt die Anweisung ON
                            zur zweiten Zeilennummer der Liste (30).
25 PRINT "DURCHGEFALLEN": GOTO 10
30 PRINT "ZU HOCH": GOTO 10
50 PRINT "ZU NIEDRIG": GOTO 10
70 END
```

OPEN
****

OPEN File-#,Geräte-# [,Sekundär-Adresse] [,"File-Name,Typ,Modus"]

Mit der OPEN-Anweisung ist der COMMODORE 16 in der Lage, mit Geräten wie der Datassette, der Diskettenstation, einem Drucker oder auch dem Bildschirm, Daten auszutauschen. Dem Wort OPEN folgt eine logische File-Nummer, auf die sich in der Folge alle BASIC-Anweisungen beziehen müssen. Diese Zahl kann zwischen 1 und 255 liegen.

Die folgende zweite Zahl ist die Gerätenummer (Primär-Adresse). Beim COMMODORE 16 sind:

0	Tastatur	
1	Datassette	(Kassettenaufzeichnung)
3	Bildschirm	
4	Drucker	(Normal-Konfiguration)
8	Diskettenlaufwerk	(Normal-Konfiguration)

Es empfiehlt sich, die Gerätenummer auch als File-Nummer zu verwenden - es merkt sich leichter.

Der zweiten Zahl kann noch eine dritte folgen: die Sekundäradresse. Bei Verwendung der Datassette bewirkt Sekundäradresse 0: Daten lesen, Sekundäradresse 1: Daten schreiben, und Sekundäradresse 2: Daten schreiben mit EOT-Markierung (End Of Tape = Bandende) am Ende. Bei Verwendung der Diskette steht die Sekundäradresse mit der Kanalnummer in Zusammenhang. Bei Druckern wird per Sekundäradresse der Druckmodus bestimmt. Mehr Informationen über die Sekundäradresse bringen die einzelnen Gerätehandbücher.

In Anschluß an die dritte Zahl kann noch ein String folgen, der sowohl ein Diskettenkommando, oder auch der Name des Files auf Kassette oder Diskette sein kann. Direkt vor dem Filenamen kann die Laufwerksnummer, gefolgt von einem Doppelpunkt, stehen. Davor noch zeigt ein 'Klammeraffe' (auf DIN-Tastatur ein §) an, daß ein evtl. schon existierendes File gleichen Namens zu überschreiben ist. Typ und Modus beziehen sich ausschließlich auf die Diskette. File-Typen bzw. Modi sind:

PRG = Programm-File
SEQ = Sequentielles File
REL = Relatives File
USR = User-File

READ = File zum Lesen eröffnen
WRITE = File zum Schreiben eröffnen
APPEND = File zum Schreiben, und zwar hinter den letzten bereits bestehenden Datensatz, eröffnen

BEISPIELE:

10 OPEN 3,3	Bildschirm als Gerät öffnen
10 OPEN 1,0	Tastatur als Gerät öffnen
20 OPEN 1,1,0,"HINAUF"	Kassette als Gerät öffnen und lesend nach dem File mit Namen "HINAUF" suchen
OPEN 4,4	Öffnet einen Kanal zum Drucker
OPEN 15,8,15	Öffnet den Befehlskanal zur Diskette
5 OPEN 8,8,12,"§0:TESTFILE,S,W"	Eröffnet ein sequentielles File zum Überschreiben auf Diskette

Siehe auch die Anweisungen: CLOSE, CMD, GET#, INPUT# und PRINT#, sowie die System-Variablen ST, DS und DS$.

PAINT
*****

PAINT [Farbregister-#] [,[a,b] [,Modus]]

```
Farbregister-# .. (0-3); Standard = 1 = Vordergrund
a,b ............. Koordinaten, skaliert (Standard = PC Position)
Modus ........... 0 = gleiche Farbe wie gewähltes Farbregister
                  1 = irgend eine Nicht-Hintergrundfarbe
```

Mit der PAINT-Anweisung lassen sich (geschlossene!) Flächen farbig gestalten. Gefärbt wird diejenige Fläche, in der die angegebenen Koordinaten liegen. Als Grenzen sind Bereiche gleicher Farbe (oder irgend eine Nicht-Hintergrundfarbe, in Abhängigkeit vom gewählten Modus) anzusehen. Am Ende des PAINT-Vorgangs befindet sich der PC wieder im Ausgangspunkt.

WICHTIG: Befindet sich der Startpunkt bereits in der Farbe, die Sie auswählten (oder in einer Nicht-Hintergrundfarbe, wenn Modus 1 gewählt wurde), dann merken Sie keine Veränderung.

BEISPIEL:

10 CIRCLE , 160,100,65,50	Zeichnet den Umriß eines Kreises.
20 PAINT , 160,100	Füllt den Kreis mit Farbe.

POKE
****

POKE Zieladresse, Wert

Mit der POKE-Anweisung kann jeder Wert im RAM-Speicherbereich des COMMODORE 16 und viele der COMMODORE 16-Input/Output-Register geändert werden. Es wird also ein Byte neu gesetzt. Der POKE-Anweisung folgen stets zwei Zahlen (oder Formeln).

Die erste Zahl bezieht sich auf eine Adresse im Speicher des Computers und kann einen Wert von 0 bis 65535 annehmen. Der Wert der zweiten Zahl liegt im Bereich von 0 bis 255 und kann an die zuvor spezifizierte Adresse geschrieben werden; der bisher dort gespeicherte Wert wird dadurch überschrieben.

BEISPIEL:

10 POKE 28000,8	Setzt Inhalt der Adresse 28000 auf 8
20 POKE 28*1000,27	Setzt Inhalt der Adresse 28000 auf 27

WICHTIG: Die Anweisung PEEK, das Gegenstück zu POKE, finden sie bei den FUNKTIONEN erklärt.

PRINT
*****
PRINT String/Variable/ [,...]

Die PRINT-Anweisung ist die wichtigste Ausgabe-Anweisung in BASIC. Obwohl die PRINT-Anweisung eine der ersten BASIC-Anweisungen ist, die gelernt wird, bleiben noch viele zusätzliche Anwendungsmöglichkeiten offen. Nach dem Wort PRINT kann stehen:

Texte innerhalb von Anführungszeichen	("Text Zeilen")
Variablen-Namen	(A, B, A$, X$)
Formeln	(SIN(23)-ABS(3*X))
Satzzeichen	(; ,)

Die Texte innerhalb der Anführungszeichen werden wortwörtlich wiedergegeben, d.h. sie werden genau so ausgedruckt, wie sie da stehen. Variablennamen werden mit ihrem Wert (gleichgültig, ob Zahlenwert oder String) ausgedruckt. Auch bei den Formeln wird nur das Ergebnis ausgegeben. Mit Hilfe der Satzzeichen kann das Format der Ausgabe am Bildschirm angepaßt werden.

Das Komma teilt den Bildschirm für den Ausdruck der Daten in vier Bereiche von je 10 Spalten Breite, während bei Verwendung des Semikolons die Daten ohne Zwischenraum ausgedruckt werden. Beide Satzzeichen können das Ende einer Anweisung bilden, was zur Folge hat, daß die nächste PRINT-Anweisung mit oder ohne Abstand (abhängig von Komma oder Semikolon) zur vorhergegangenen direkt anschließend druckt.

BEISPIELE:

	Ausdruck - Ergebnis
10 PRINT "HALLO"	HALLO
20 A$="IHR ALLE":PRINT"HALLO,"A$	HALLO,IHR ALLE
30 A=4:B=2:PRINT A+B	6 (Leerstelle für Vorzeichen)
50 J=41:PRINT J;:PRINT J-1	41 40 (Leerstelle für Vorzeichen)
60 C=A+B:D=A-B:PRINT A;B;C,D	4 2 6 2

Siehe auch die FUNKTIONEN: POS(), SPC(), und TAB()

PRINT#
******

PRINT# File-#, String/Variable/ [, ...]

Zwischen der Anweisung PRINT# und der vorher beschriebenen Anweisung PRINT besteht ein kleiner Unterschied. Auf das Wort PRINT# folgt stets eine Zahl, die mit dem vorher geöffneten Gerät oder File (s. OPEN) korrespondiert. Dieser Zahl folgt ein Komma und dann die Liste der zu druckenden Strings bzw. Variablen. In dieser Liste bewirkt das Semikolon das gleiche, wie oben bei PRINT beschrieben. VORSICHT! Nicht alle Geräte arbeiten mit den Funktionen Komma (wirkt wie 10 Leerzeichen) und TAB! Den jeweiligen Gerätehandbüchern ist zu entnehmen, wie diese auf Steuerzeichen reagieren, die mit dem PRINT#-Befehl (z.B. in Form eines Strings) gesendet werden.

BEISPIELE:

```
100 PRINT#1, "HALLO IHR ALLE! ",A$,B$,
110 PRINT#1, A ; "," ; B ; "," ; C
```

Ausgabe von Kommata zwischen Zahlenwerten, damit diese von einer INPUT#-Anweisung gelesen werden können.

PRINT USING
****

PRINT USING "Formatvorgabe"; String/Variable [,...]

Die Anweisung PRINT USING ermöglicht es, das Format von Strings oder Zahlenreihen vorzugeben, die auf den Bildschirm, den Drucker oder ein anderes Gerät ausgegeben werden sollen.

Das gewünschte Format steht zwischen Anführungszeichen und wird Formatvorgabe genannt. Daran anschließend folgt ein Semikolon und die Liste der auszudruckenden Strings oder Variablen. Statt der Variablen können ebenso die aktuellen Werte stehen, die ausgegeben werden sollen.

BEISPIELE:

```
5 X=32: Y=100.23: A$="COMPUTER 16"
10 PRINT USING "$##.##";13.25,X,Y
20 PRINT USING "###>#";"CBM",A$
```

Geben Sie nun RUN ein und drücken die Taste <Return>, so wird entsprechend den Angaben in Zeile 10 gedruckt:

$13.25	$32.00	$*****	Die Sterne (*****) werden anstelle des Y-Wertes gedruckt, weil der Wert Y fünf Dezimalstellen umfaßt und dies mit der Formatanweisung von vier Dezimalstellen "$##.##" nicht in Einklang gebracht werden kann.

Nach den Angaben in Zeile 20 wird gedruckt:

CBM	COMMODORE 16	Bevor der String 'CBM' gedruckt wird, werden drei Leerstellen, wie im Formatstring vorgegeben "###>#", gedruckt.

Welche Zeichen beeinflussen was?

Zeichen	Variable	String
Nummern-Raute (#)	X	X
Plus (+)	X	
Minus (-)	X	
Dezimalpunkt (.)	X	
Komma (,)	X	
Dollarzeichen ($)	X	
Vier Exponentzeichen (↑↑↑↑)	X	
Gleichheitszeichen (=)		X
Größer-als-Zeichen (>)		X

Die Nummern-Raute (#) reserviert je Einzelzeichen einen Platz im Ausgabefeld. Enthält die Datenliste mehr Zeichen, als Rauten (#) im Formatstring vorgegeben sind, dann passiert folgendes:

Statt mit Zahlen wird das vorgegebene Datenfeld mit Sternen (*) gefüllt (siehe auch Zeile 10 im obigen Beispiel).

BEISPIEL:

```
10 PRINT USING "####";A
```

Für die nachstehenden Werte von A erfolgt der Ausdruck:

A = 12.34	12
A = 567.89	568
A = 123456	****

Bei einem STRING werden nur so viele Stellen gedruckt, wie Rautenzeichen (#) dafür vorgegeben sind; der Rest wird rechts abgeschnitten.

Ein Plus(+)- oder ein Minus(-)-Zeichen kann sowohl an erster wie an letzter Stelle des Formatfeldes stehen - jedoch nicht an beiden Stellen. Steht das Pluszeichen im Formatfeld, so wird es bei positiven, das Minuszeichen bei negativen Zahlen gesetzt.

Steht im Formatfeld ein Minuszeichen, so wird bei positiven Werten ein Leerzeichen, bei negativen das Minuszeichen ausgegeben.

Geben Sie im Formatstring weder ein Plus- noch ein Minuszeichen für einen Zahlenwert an, so wird unmittelbar vor der ersten Stelle ein Minuszeichen oder ein Dollarzeichen gedruckt, wenn die Zahl negativ ist. Ist die Zahl positiv, wird kein Vorzeichen gedruckt - dies bedeutet, daß Ihnen bei positiven Zahlen eine Stelle mehr zur Verfügung steht. Existieren mehr Stellen, als mit den Symbolen '#' bzw. '+' oder ' ' vorgegeben, dann tritt ein 'OVERFLOW' (= Kapazitätsüberschreitung) ein, und das Druckfeld wird mit Sternen (*) gefüllt.

Der Dezimalpunkt (.) im Formatfeld bestimmt die Position des Dezimalpunkts im auszudruckenden Zahlenfeld. Pro Formatstring darf nur ein Dezimalpunkt fixiert werden. Wird im Formatstring kein Dezimalpunkt festgelegt, wird der ausgedruckte Zahlenwert auf die nächstgelegene Zahl gerundet und ohne Dezimalstellen ausgedruckt.

Achten Sie bei der Festlegung des Dezimalpunkts auf die Anzahl der zu druckenden Stellen (einschließlich Minuszeichen, wenn es sich um einen negativen Wert handelt), ob sie sich mit der Anzahl der Rauten (#) vor dem Dezimalpunkt decken. Bei zu vielen Stellen tritt wieder ein OVERFLOW ein, und das Druckfeld wird mit Sternchen (*) gefüllt.

Mit einem Komma im Formatfeld läßt sich ein Komma auch im numerischen Feld plazieren, wobei sich die Stelle im Formatstring mit der Stelle im Ausdruck deckt. Es werden nur Kommata innerhalb einer Zahl gedruckt. Unbenützte Kommata vor der ersten Stelle erscheinen als Füllzeichen. Vor dem ersten Komma im Formatfeld muß mindestens eine Raute (#) stehen.

Werden Kommata in einem Feld definiert und die Zahl ist negativ, dann wird als erstes Zeichen ein Minus (-) gedruckt, auch wenn diese Position durch ein Komma im Formatfeld besetzt wurde.

BEISPIELE:

Formatfeld	Zahlenwert	Ausdruck	Erklärung
##.#+	-.01	0.0-	Die erste Null wird eingefügt, die Eins wird abgerundet.
##.#-	1	1.0	Die Nachkomma-Null wird angefügt.
####	-100.5	-101	Gerundet ohne Kommastellen.
####	-1000	****	OVERFLOW, weil vier Stellen und das Minuszeichen nicht mehr in das Druckfeld passen.
###.	10	10.	Der Dezimalpunkt wird angefügt.
#$##	1	$1	Vorangestelltes Dollarzeichen.

Ein Dollarzeichen ($) im Formatfeld bestimmt, daß auch ein Dollarzeichen mit der Zahl gedruckt wird. Soll sich das Dollarzeichen der wechselnden Stellenzahl anpassen (stets vor der Zahl stehen), dann muß vor dem Dollarzeichen noch eine Raute (#) plaziert werden. Fehlt diese davorstehende Raute (#), dann wird das Dollarzeichen entsprechend seiner Stelle im Formatfeld gedruckt.

Positionieren Sie Kommata und/oder ein Plus- oder ein Minuszeichen in einem Formatstring, in dem auch ein Dollarzeichen enthalten ist, so wird vom Programm ein Komma oder ein Plus/Minuszeichen vor das Dollarzeichen gedruckt.

Die vier Exponentenpfeile (↑↑↑↑) symbolisieren, daß die Zahl im wissenschaftlichen Format (E-Format) ausgedruckt wird. Zusätzlich zu den ↑↑↑↑ ist mit Rauten (#) die Stellenzahl (Feldbreite) zu definieren. Die ↑↑↑↑ können sowohl vor wie auch nach den Rauten im Feldstring plaziert werden. Eine Eingabe von mehr als einem Pfeil (↑) und weniger als vier Pfeile führt zu einem SYNTAX ERROR. Werden mehr als vier Pfeile fixiert, so gelten nur die ersten vier. Weitere Pfeile werden ignoriert.

Mit dem Gleichheitszeichen (=) wird ein String innerhalb des Formatfeldes zentriert. Die Breite des Feldes geben Sie mit der Anzahl der Zeichen # und = im Feldstring an. Enthält der String weniger Zeichen als im Feld angegeben, so wird er im Feld zentriert. Enthält der String mehr Zeichen, als in das definierte Feld passen, werden die überzähligen rechts abgeschnitten und das Feld vollständig ausgefüllt.

Das 'Größer-als' - Zeichen (>) dient dazu, den String im Feld rechtsbündig zu justieren. Die Breite des Feldes geben Sie mit der Anzahl der Zeichen # und = im Feldstring an. Enthält der String weniger Zeichen als im Feld angegeben, so wird er im Feld rechtsbündig justiert. Enthält der String mehr Zeichen, als in das definierte Feld passen, werden die überzähligen rechts abgeschnitten und das Feld vollständig ausgefüllt.

PUDEF
*****
<u>PUDEF</u> "1 bis 4 Zeichen"

Mit der Anweisung PUDEF lassen sich bis zu vier Zeichen in der PRINT USING-Anweisung neu definieren. Sie können Leerstellen, Kommata, Dezimalpunkte und Dollarzeichen durch andere Zeichen ersetzen, indem Sie das neue Zeichen an eine der vier möglichen Stellen in der PUDEF-Anweisung plazieren.

Position 1 Repräsentiert die Leerstelle. Standard ist ein 'BLANK' (' ' = Leerstelle). Sie schreiben an diese erste Stelle jenen Buchstaben, der sich nach Durchführung der Anweisung an Stelle der Leerzeichen im String befinden soll.

Position 2 Steht für die Kommata. Standard ist ein Komma (,).

Position 3 Wird vom Dezimalpunkt eingenommen.

Position 4 Gilt für das Dollarzeichen.

BEISPIELE:

10 PUDEF "*" Anstelle der Leerstellen werden Sterne (*) gedruckt.

20 PUDEF " &" Anstelle der Kommata werden 'kaufm. und' (&) gedruckt.

30 PUDEF " .," Anstelle der Kommata werden Dezimalpunkte und anstelle der Dezimalpunkte Kommata gedruckt.

40 PUDEF " .,£" Anstelle des Dollarzeichens wird das Englische Pfund, anstelle der Kommata werden Dezimalpunkte und anstelle der Dezimalpunkte Kommata gedruckt.

Achten Sie auf genaue Eingabe, weil der COMMODORE 16 sonst unerwünschte Zeichen anstelle der Standardwerte ausdruckt.

READ
****
READ Variablen-Liste

Die Anweisung READ wird benutzt, um in DATA-Zeilen gespeicherte Information in Variable zu übertragen. Die Variablenliste der READ-Anweisung kann sowohl Strings als auch numerische Variablen enthalten. Zu beachten ist jedoch, daß kein String eingelesen wird, wenn die READ-Anweisung einen numerischen Wert erwartet; dies führt zu einer Fehlermeldung.

BEISPIEL:

```
READ A$, G$, Y
```

REM
***
REM Bemerkung

Mit Hilfe der Anweisung REM (von REMark = Bemerkung) können an jeder Stelle des Programmlistings Anmerkungen notiert werden. Dadurch lassen sich Teile des Programms erklären, Informationen über den Programmierer festhalten, usw. REM-Anweisungen haben keinen Einfluß auf den Programmablauf, mit der Ausnahme, daß das Programm länger und dadurch langsamer wird. Dem Wort REM kann jede Art Text folgen, aber auch Grafikzeichen (bzw. Großbuchstaben), die jedoch, sofern sie nicht in Anführungsstriche eingeschlossen sind, beim LIST-Befehl als Schlüsselworte ausgegeben werden. Die REM Anweisung muß die letzte Anweisung einer Zeile sein: nach REM wird kein Befehl mehr ausgeführt, auch nicht nach einem Doppelpunkt!

BEISPIEL:

```
10 NEXT X: REM SCHLEIFEN-VARIABLE X ZAEHLT DURCHLAEUFE
```

RESTORE
****
<u>RESTORE</u> [Zeilen-Nummer]

Mit Hilfe der Anweisung RESTORE ist es möglich, DATA-Zeilen mehrfach zu lesen. Beim Abarbeiten dieser Anweisung wird der DATA-Zeiger wieder auf die erste Position der ersten DATA-Zeile gesetzt. Folgt der RESTORE-Anweisung eine Zeilennummer, dann wird der DATA-Zeiger auf diese Zeile gesetzt.

BEISPIEL:

125 RESTORE 200

RESUME
******
<u>RESUME</u> [Zeilen-Nummer / <u>NEXT</u>]

Um nach einer Fehleranalyse mit der Anweisung TRAP (siehe dort) den Programmablauf fortzusetzen, wird die Anweisung RESUME gegeben. Folgen der Anweisung keine Parameter, dann fährt das Programm mit dem Befehl fort, welcher den Fehler verursachte. Mit RESUME NEXT setzt das Programm mit dem Befehl hinter demjenigen, der den Fehler erzeugte, fort. RESUME mit einer Zeilen-Nummer funktioniert wie eine GOTO-Anweisung und veranlaßt im Fehlerfall die Programmfortführung ab der angegebenen Zeile.

RETURN
******
RETURN

Die RETURN-Anweisung wird immer in Verbindung mit der GOSUB-Anweisung benützt. Erreicht der Programmablauf eine RETURN-Anweisung, verzweigt das Programm zu dem der zugehörigen GOSUB-Anweisung folgenden Befehl. Existiert keine zugehörige GOSUB-Anweisung, dann kommt die Fehlermeldung RETURN WITHOUT GOSUB ERROR, und das Programm wird gestoppt.

SCALE
*****
SCALE <1/0>

Mit Hilfe der Anweisung SCALE kann im Mehrfarben-Modus und im Hi-Res-Modus die Skalierung der 'Bit Maps' geändert werde.

SCALE 1 schaltet die Skalierung ein. Statt der Koordinatenwerte in nachstehender Tabelle stehen dann in beiden Achsen (X und Y) je die Werte von 0 bis 1023 zur Verfügung. Die Normalwerte lauten hingegen:

Mehrfarben Modus	X = 0 bis 159	Y = 0 bis 199
Hochauflösende Grafik Modus (Hi-Res)	0 bis 319	0 bis 199
Geteilter Bildschirm (beide Modi)	wie oben	0 bis 159

Mit SCALE 0 wird die höhere Skalierung wieder abgeschaltet.

SCNCLR
******
SCNCLR

Mit der Anweisung SCNCLR wird der Bildschirm, unabhängig vom Modus, gelöscht.

SOUND
*****
SOUND (Stimme, Notenwert, Klangdauer)

Die Anweisung SOUND erzeugt Töne bzw. Geräusche mit einer von zwei Stimmen, wobei der Bereich der Notenwerte von 0 bis 1023 reicht und die Klangdauer zwischen 0 und 65535 Fünfzigstel einer Sekunde (= 1311 Sekunden = 21.8 Minuten) erreichen kann.

Stimme #	Ergibt
1	Einzelton (1. Stimme)
2	Einzelton (2. Stimme)
3	Geräusch (2. Stimme)

Wird ein SOUND für die Stimme N aufgerufen, und dieselbe Stimme N erzeugt gerade noch einen Ton, dann wartet das BASIC-Programm mit dem Programmablauf, bis der gegenwärtige Ton ausgeklungen ist. Einen Spezialfall stellt SOUND mit der Klangdauer 0 dar. Damit wird nämlich der gegenwärtig erzeugte Ton abrupt abgebrochen, unabhängig von der eingegebenen Klangdauer. Die Zuordnung der einzelnen Notenwerte zu den tatsächlichen Tonhöhen finden Sie im Anhang dieses Handbuchs.

BEISPIEL:

SOUND 2, 800, 3000 Spielt Stimme 2 mit einem Notenwert 800 eine Minute lang.

SSHAPE/GSHAPE
****
SSHAPE/GSHAPE

SSHAPE- und GSHAPE-Anweisungen werden eingesetzt, um Rechteckflächen, die im Mehrfarben- oder Hi-Res-Modus dargestellt sind, als BASIC-Stringvariable abzuspeichern bzw. wieder zu laden. Der Speicherbefehl lautet:

SSHAPE String[-Variable], a1,b1 [,a2,b2]

```
String[-Variable] .... Variable, in die der String abgelegt wird
a1, b1 ............... Eckkoordinate (skaliert)
a2, b2 ............... a1, b1 gegenüberliegende Eck-
                       koordinate (Standard = PC)
```

Da BASIC die Länge einer Stringvariablen mit 255 Zeichen begrenzt, ist die Größe des abzuspeichernden Bereichs begrenzt. Die pro Fläche notwendige Variablen-Länge läßt sich mit nachstehenden (unskalierten) Formeln berechnen:

```
L(MFM)=INT((ABS(a1-a2)+1)/4+.99)*(ABS(b1-b2)+1)+4
L(HIR)=INT((ABS(a1-a2)+1)/8+.99)*(ABS(b1-b2)+1)+4
```

MFM = MehrFarbenModus   HIR = Hochauflösende Grafik

Die Fläche wird Zeile für Zeile abgespeichert. Die letzten vier Bytes des Strings enthalten jeweils die Spalten- und Zeilenlänge abzüglich Eins (d.h. ABS(a1-a2)) im Low/High-Byte-Format (niederwertiges Byte zuerst, dann das höherwertige Byte (wie beim Programmieren in Maschinensprache). Befinden Sie sich im skalierten Modus, dann muß die Länge in der X-Achse durch 3.2 und in der Y-Achse durch 5.12 dividiert werden.

GSHAPE String[-Variable] [, [a,b] [,Modus]]

Mit der Anweisung GSHAPE läßt sich eine abgespeicherte Fläche wieder an jede Stelle des Bildschirms laden.

String[-Variable] ... Enthält wiederzugebenden String

a,b .... Linke, obere Ecke, ab der gezeichnet werden soll (Standard = PC), skaliert

Modus .... Wiedergabe-Modus:
0: Wiedergabe wie aufgenommen (Standard)
1: Wiedergabe in REVERSE-Modus
2: Wiedergabe mit Fläche ODER-verknüpfen
3: Wiedergabe mit Fläche UND-verknüpfen
4: Wiedergabe mit Fläche Exclusiv-ODER-verknüpfen

BEISPIELE:

SSHAPE "SCHIFF", 0, 0	Speichert Bildschirmbereich ab der linken oberen Ecke bis zur CURSOR Position unter dem Namen "SCHIFF".
GSHAPE "SCHIFF",,,1	Gibt die abgespeicherte Fläche namens "SCHIFF" mit umgekehrten Hinter- und Vordergrund-Farben ab der CURSOR-Position wieder.

STOP
****
STOP

Die Anweisung STOP hält den Programmablauf an dieser Stelle mit der Meldung BREAK IN LINE # an, wobei die Line # die Programmzeile ist, in der die STOP-Anweisung steht. Mit der Anweisung CONT läßt sich das Programm ab dem der Anweisung STOP folgenden Befehl wieder starten. Die Anweisung STOP wird hauptsächlich bei der Fehlersuche in BASIC--Programmen eingesetzt.

SYS
***
SYS Speicher-Adresse

Dem Wort SYS folgt als Ganzzahl eine Dezimalzahl oder eine Variable im Bereich von 0 bis 65535. Mit der SYS-Anweisung wird ein Maschinensprache-Programm ab der angegeben Speicheradresse gestartet. Die Anweisung ist ähnlich der USR-Anweisung, mit dem Unterschied, daß kein Parameter übergeben wird. Das Maschinenprogramm muß mit einem RTS-Befehl ($60) enden.

TRAP
****
TRAP [Zeilen-#]

Mit der Anweisung TRAP läßt sich innerhalb eines Programms auf alle Fehlersituationen (einschließlich der Taste <Run/Stop> und mit Ausnahme der Fehlermeldung UNDEF'D STATEMENT ERROR) gezielt reagieren. Im Falle eines aufgetretenen Fehlers wird das Fehler-FLAG gesetzt, und die Programmweiterführung wird zu der in der TRAP Anweisung gegebenen Zeilennummer umgeleitet. Die Zeilennummer, in welcher der Fehler auftrat, ist in der Variablen EL gespeichert. Die eigentliche Fehlermeldung enthält die System Variable ER. Mit der String-Funktion ERR$(ER) ist jede Fehlermeldung zur korrespondierenden Fehlersituation ER auslesbar.

WICHTIG: Ein Fehler in einer TRAP-Programmroutine kann nicht ausgelesen (ge'trappt') werden. Mit der Anweisung RESUME kann der Programmablauf wieder aufgenommen werden. Ohne Angabe der Zeilennummer wird die Fehlerverfolgung wieder ausgeschaltet.

TRON
****
TRON

Die Anweisung TRON dient der Fehlersuche. Ab dieser Anweisung wird Abarbeitung des Programms protokolliert. Dabei wird jede Anweisung ausgeführt und die Zeilennummer dieser Anweisung ausgegeben.

TROFF
*****
TROFF

Mit der Anweisung TROFF wird die Fehlersuche (TRON) wieder abgeschaltet.

VOL
***
VOL Lautstärke-Pegel

Mit der Anweisung VOL und einem Parameter zwischen 0 und 8 wird die Lautstärke der im COMMODORE 16 erzeugten Töne und Geräusche eingestellt. Maximale Lautstärke wird mit 8 erreicht, mit 0 wird abgeschaltet. Mit der Anweisung VOL werden beide Stimmen gesteuert.

WAIT
****
<u>WAIT</u> Speicher Adresse, Wert 1 [, Wert 2]

Mit der Anweisung WAIT kann ein Programm angehalten werden, und zwar solange, bis der Speicherinhalt einer vorgegebenen Speicheradresse einen bestimmten Wert erreicht hat, bzw. sich in einer speziellen Art verändert hat. Der Speicheradreßbereich liegt zwischen 0 und 65535. Die Werte 1 und 2 können von 0 bis 255 reichen.

Der Inhalt der Speicherstelle wird zuerst mit dem Wert 2 (falls vorgegeben) Exclusiv-ODER-verknüpft und anschließend mit Wert 1 logisch UND-verknüpft. Ergibt dies ein Resultat von Null, wird der Inhalt der Speicherstelle erneut überprüft. Ist das Ergebnis nicht Null, setzt das Programm bei der nächsten Anweisung fort.

VORSICHT! Nimmt der Inhalt der angesprochenen Speicheradresse nicht den erwarteten Wert an, dann kann der Rechner nur durch einen RESET wieder zum Leben erweckt werden. Gehen Sie deshalb sorgfältig mit diesem Befehl um! Verwenden Sie ihn nur zur Abfrage von Ein-/Ausgabe-Registern, Tastaturabfragen und ähnlichem.

```
*****************************************************
*  WEITERE INFORMATIONEN ZU DEN GRAFIK-ANWEISUNGEN  *
*****************************************************
```

Es gibt ein paar Hinweise, die für alle Anweisungen gültig sind, die sich mit Hochauflösender Grafik beschäftigen. Zuerst Hinweise zum Pixel-Cursor (PC). Dieser PC ist dem regulären CURSOR im Text-Modus ähnlich; er bestimmt die nächste Position, an der ein weiteres Pixel gesetzt wird. Im Gegensatz zum normalen CURSOR ist der PC nicht sichtbar. In allen Zeichen-Anweisungen wird der PC verwendet. Zusätzlich läßt sich der PC mit der LOCATE-Anweisung überallhin positionieren - ohne dazwischen etwas zu zeichnen.

Überall, wo (X,Y)-Koordinaten in einer Zeichen-Anweisung vorkommen, können stattdessen auch RELATIVE Koordinaten verwendet werden. Relative Koordinaten bauen auf der momentanen Position des PC auf. Um die relativen Koordinaten zu verwenden, genügt es, vor die eigentlichen Koordinaten ein Plus- oder Minuszeichen zu setzen. Danach bewegt ein Pluszeichen vor der X-Koordinate den PC nach rechts. Ein Minus vor der X-Koordinate bewegt den PC nach links. Ganz ähnlich bewegt ein Minuszeichen vor der Y-Koordinate den PC nach oben und ein Plus nach unten.

Ein Beispiel:

LOCATE +100,-25	Bewegt den PC 100 Pixel nach rechts und 25 nach oben.
DRAW,+10,+10TO100,100	Zeichnet eine Linie 10 Pixel rechts und 10 Pixel unter der gegenwärtigen PC-Position zu dem Punkt mit den absoluten Werten 100, 100.

Auf gleiche Weise lassen sich Abstände und Winkelmaße RELATIV zur gegenwärtigen PC-Position festlegen, indem die beiden Parameter durch ein Semikolon (;) getrennt werden.

Ein Beispiel:

LOCATE 50;45	Bewegt den PC aus seiner momentanen Position um 50 Punkte und unter einem Winkel von 45 Grad weg.

```
**************
* FUNKTIONEN *
**************

**************************
* NUMERISCHE FUNKTIONEN *
**************************
```

Numerische Funktionen werden so bezeichnet, da sie numerische Zahlenwerte ergeben. Der Bereich, den sie abdecken, reicht von mathematischen Rechenfunktionen bis zur Bestimmung einer Bildschirmposition. Numerische Funktionen folgen nachstehenden Regeln:

FUNKTION (Argument),

wobei das Argument je nachdem ein numerischer Wert, eine Variable oder ein String (Zeichenkette) sein kann. Es können mehrere Argumente, auch unterschiedlichen Typs, erforderlich sein.

ABS (X) (Absoluter Wert)

Die Funktion ABS gibt den absoluten Wert einer Zahl an; d.h. ohne Berücksichtigung eines Vorzeichens (+ oder -). Das Ergebnis ist stets Positiv bzw. Null.

ASC (X$)

Mit der Funktion ASC wird der ASCII-Kode (Zahlenwert, CBM-Version, siehe Anhang) des ersten Zeichens im String X$ ausgegeben.

ATN (X) (Arcustangens)

Ergibt den Winkel (in Bogenmaß, siehe SIN), dessen Tangens gleich X ist.

COS (X) (Cosinus)

Ergibt den Cosinus des Winkels X. Der Winkel ist im Bogenmaß einzugeben (siehe SIN).

DEC (H$) (H$ für 'Hexadezimal-String')

Liefert den Dezimal-Wert eines Hexadezimal-Strings (0 < Hexadezimal-String < FFFF , zugelassene Zeichen sind 0-9 und A-F).

BEISPIEL:

N = DEC ("F4") ergibt den Wert 244 .

EXP (X) (Exponential- oder e-Funktion)

Ergibt die X-te Potenz der mathematischen Konstanten e (e=2.71828183).

FNxx (X)

Berechnet den Wert der Benutzer-definierten Funktion FNxx, die in einer DEF FNxx Anweisung festgelegt wurde.

INSTR (String 1, String 2 [, Start-Position])

Bringt die Position, ab der String 2 in String 1 enthalten ist. Die Suche beginnt ab der Start-Position. Standardmäßig befindet sich die Start-Position am Beginn des ersten Strings. Wird keine Deckungsgleichheit gefunden, ist der Wert 0.

BEISPIEL:

PRINT INSTR ("DIE KATZE IM SACK", "KATZE")

bringt als Ergebnis 5, weil der String (2) KATZE sich ab der fünften Stelle mit dem entsprechenden Teil von String (1) deckt.

INT (X)

Ergibt den ganzzahligen Anteil von X. Das Ergebnis ist immer kleiner oder gleich X. Das bedeutet, daß bei positiven Zahlen alle Stellen nach dem Dezimalpunkt abgeschnitten werden, daß aber negative Zahlen dem Betrag nach größer werden (z.B. INT(-4.5)= -5).

Wird die INT-Funktion zur Rundung eingesetzt, dann lautet die Anweisung INT(X + .5).

BEISPIEL:

X = INT(X*100+.5)/100 — Damit wird auf den nächst höheren 1/100-Wert (z.B. Pfennig) gerundet.

JOY (n)

Für den Joystick 1 ist n = 1
Für den Joystick 2 ist n = 2

Jeder Wert größer 127 bedeutet, daß auch der Feuerknopf gedrückt ist. Die einzelnen Richtungen ergeben sich aus nachstehender Matrix:

```
                                 RAUF
FEUER = 128 +
*****                              1
                           8               2
           LINKS  7                0                3  RECHTS
                           6               4
                                   5
                                 RUNTER
```

BEISPIEL:

Die Abfrage von JOY(2) liefert 135, wenn am Joystick #2 gleichzeitig der Feuerknopf gedrückt und der Hebel nach links gehalten wird.

LOG (X)

Ergibt den natürlichen Logarithmus (zur Basis e) vom Wert X. Zur Umwandlung in den Briggsschen Logarithmus (dekadischer Logarithmus, zur Basis 10) wird durch LOG(10) dividiert.

<u>PEEK</u> (X)

Diese Funktion ergibt den Inhalt der Speicheradresse X, wobei die Adresse X zwischen 0 und 65535 liegen muß. Der Inhalt kann Werte von 0 bis 255 annehmen. PEEK wird oft im Zusammenhang mit der Anweisung POKE eingesetzt.

<u>RCLR</u> (N)

Gibt die der Farbquelle N (0 < N < 4) zugeordnete, gegenwärtige Farbzone an. (0= Bildschirm-Hintergrund, 1= Vordergrund, 2= Mehrfarben 1, 3= Mehrfarben 2, 4= Bildschirm-Rand).

<u>RDOT</u> (N)

Gibt die momentanen Koordinaten des PC (=Pixel Cursor) an.

N = 0 für x-Position
N = 1 für y-Position
N = 2 für Farbquellen-#

<u>RGR</u> (X)

Liefert gegenwärtigen Grafik-Modus, wobei X ein 'Dummy' (= beliebiger Füllwert) ist.

<u>RLUM</u> (N)

Gibt die der Farbzone N zugewiesene Farbintensität an.

<u>RND</u> (X)

Mit dieser Funktion erhalten Sie eine Zufallszahl zwischen 0 und 1. Dies ist u.a. hilfreich bei der Programmierung von Spielen, um

Würfelwerte oder ähnliche Zufallswerte zu simulieren, oder auch wichtig für statistische Anwendungen. Die erste Zufallszahl sollte mit der Formel RND(-TI) erzeugt werden, damit bei jedem Start stets eine andere Zahl erzeugt wird. Danach kann die Zahl in der Variablen X eine Eins oder eine andere positive Zahl sein (X repräsentiert die Kernzahl, auf der die Zufallszahl aufbaut). Ist X = 0, dann wird RND von der eingebauten Uhr mit jedem Aufruf von RND auf eine Kernzahl, die sich durch die Uhr ergibt, zurückgesetzt. Ein negativer Wert für X bedeutet bei jedem RND-Aufruf eine Rückstellung der Kernzahl. Auch bei wiederholtem Aufruf der RND-Anweisung wird, bei unverändertem negativen Argument, immer die gleiche Folge von Zufallszahlen gebildet werden. Ein positiver Wert für X ergibt neue Zufallszahlen, die auf der vorangegangenen Kernzahl basieren.

Als Beispiel simulieren wir ein Würfelspiel mit der Formel INT(RND(1)*6+1). Als erstes werden damit die Zufallszahlen des COMMODORE 16, die zwischen 0 und 1 liegen, mit 6 multipliziert. Daraus entstehen Zufallszahlen, die größer als 0 und kleiner als 6 sind ($0<n<6$). Damit Zufallszahlen im Bereich $1<n<7$ entstehen, addieren wir 1, und mit der INT-Funktion werden alle Dezimalstellen abgetrennt. Somit erhalten wir Ganzzahlen zwischen 1 und 6. Zwei Würfel lassen sich simulieren, wenn zwei der durch obige Formel erhaltenen Zahlen zusammengezählt werden.

BEISPIEL:

```
100 X=INT(RND(1)*6)+INT(RND(1)*6)+2      Simuliert zwei Würfel
100 X=INT(RND(1)*1000)+1                 Erzeugt Zahlen von 1-1000
100 X=INT(RND(1)*150)+100                Erzeugt Zahlen von 100-249
```

<u>SGN</u> (X)

Mit dieser Funktion läßt sich das Vorzeichen von X ermitteln. Bei positivem X ergibt sich +1, bei Null 0 und bei negativem X -1.

<u>SIN</u> (X)

Dies ist die trigonometrische Funktion Sinus. Das Ergebnis ist der Sinus des Winkels X, wobei X in Bogenmaß einzugeben ist. Ist X im Gradmaß angegeben, so ist die Umrechnung SIN(X* π /180) zu verwenden.

<u>SQR</u> (X)

Mit der Funktion SQR (Square-Root) wird die Quadratwurzel von X gezogen, wenn X gleich (=) oder größer (>) Null (0) ist. Bei negativen Zahlen kommt die Fehlermeldung ILLEGAL QUANTITY ERROR.

<u>TAN</u> (X)

Der Tangens des Winkels X wird mit dieser Funktion berechnet. Der X-Wert wird im Bogenmaß eingegeben (siehe SIN).

<u>USR</u> (X)

Wird diese Funktion angesprochen, dann springt das Programm in ein Maschinenprogramm, dessen Startadresse in den Speicherstellen 1281 und 1282 steht. Diese Adresse muß zuvor in LO-/HI-Byte-Schreibweise (d.h. Adresse = LO+256*HI, wie in Maschinensprache üblich) mit POKE in 1281 (für LO) und 1282 (für HI) plaziert werden. Der Parameter X wird dem 'Floating Point Akkumulator' = 'FPA' übergeben. Das mit USR aufgerufene Unterprogramm (in Assembler) wertet den Inhalt des 'FPA' aus und gibt einen neuen Wert zurück.

VAL (X$)

Diese Funktion wandelt den String X$ in eine Zahl um. Sie ist besonders wichtig, wenn es um die Umkehr der Funktion STR$ (siehe String-Funktionen) geht. Der String selbst wird Zeichen für Zeichen, von links beginnend, nach rechts - bis zur vorgegebenen Gesamtstellenzahl - nach Zahlen abgefragt. Findet der COMMODORE 16 ein unzulässiges Zeichen, wird der String bis zu diesem unzulässigen Zeichen in eine Zahl konvertiert.

BEISPIEL:

```
10 X = VAL("123.456")          Ergibt  X=123.456
10 X = VAL("3E03")             Ergibt  X=3000
10 X = VAL("12A13B")           Ergibt  X=12
10 X = VAL("RIUO17*")          Ergibt  X=0
10 X = VAL("-1.23.23.23")      Ergibt  X=-1.23
```

```
***********************
*  STRING-FUNKTIONEN  *
***********************
```

String-Funktionen liefern immer einen String (Zeichenkette) als Ergebnis. Als Argumente können Zahlen oder Strings auftreten.

CHR$ (X)

Die Funktion CHR$ (= Umkehrfunktion zu ASC(X$)) bildet einen String der Länge eins (d.h. ein Zeichen), dessen ASCII-Kode (CBM-Version, siehe Anhang) gleich X ist.

ERR$ (N)

Zeigt den String an, der den Fehlerzustand N (siehe TRAP) beschreibt.

HEX$ (N)

Diese Funktion liefert den vier Zeichen langen Hexadezimalwert einer ganzen Zahl N (0 < N < 65535).

LEFT$ (X$,X)

Ergibt den String, der aus den ersten X Zeichen des Strings X$ - von links gezählt - besteht.

LEN (X$)

Die Funktion LEN gibt die Gesamtlänge (einschließlich Leerstellen und Steuerzeichen) des Strings X$ als Zahl aus, ist also eigentlich eine Numerische Funktion.

<u>MID$</u> (X$,S,X)

Diese Funktion ergibt einen String, der ab dem S-ten Zeichen eine Anzahl von X Zeichen aus dem String X$ enthält. MID$ kann sowohl auf der linken Seite einer Variablen Zuweisung (MID$ = ...) stehen, als auch als Funktion wirken.

Für das erste Zeichen gilt S=1. Bei fehlender Längenangabe X besteht der Teilstring aus allen Zeichen ab dem S-ten Zeichen. X bzw. S sind auf einen Wertbereich von 0 bis 255 beschränkt.

BEISPIEL:

```
10 A$ = "ZUM LETZTEN MAL":
20 PRINT A$
30 MID$(A$,5,4)=" ERS"
40 PRINT A$                         Ergibt 'ZUM  ERSTEN MAL'.
```

<u>RIGHT$</u> (X$,X)

Ergibt X Zeichen des Strings X$ - von rechts gezählt.

<u>STR$</u> (X)

Damit wird ein String gebildet, der der Zahl von X entspricht.

BEISPIEL:

```
X = 99
A$ = STR$ (X)
PRINT A$
RUN
 99
```

```
*************************
*  SONSTIGE FUNKTIONEN  *
*************************
```

FRE (X)

Die Funktion FRE ergibt die Anzahl freier Bytes im Speicher des COMMODORE 16. Der Wert X ist ein beliebiger 'Dummy'.

POS (X)

Teilt die derzeitige-CURSOR Position mit (0 bis 79 in einer logischen Bildschirmzeile). Da der COMMODORE 16 einen 40-Zeichen-Bildschirm besitzt, beziehen sich die Zahlen 40 bis 79 auf die jeweils zweite Bildschirmzeile. Der Wert X ist ein beliebiger 'Dummy'.

SPC (X)

Diese Funktion wird in Verbindung mit der PRINT-Anweisung verwendet, um X Stellen zu überspringen. Dabei kann X den Wert von 0 bis 255 annehmen.

TAB (X)

Auch diese Funktion wird mit PRINT-Anweisungen verwendet. Das nächste Zeichen wird in Spalte X gedruckt. Der Wert X kann zwischen 0 und 255 liegen. Ist auf Druckern nicht anwendbar.

π (= PI)

Das PI-Symbol hat innerhalb einer Gleichung den fest gespeicherten Wert von 3.14159265.

****
* VARIABLEN UND OPERATOREN *
****

****
* VARIABLEN *
****

Der COMMODORE 16 kennt drei Variablen-Arten in BASIC. Es sind dies: Normale Gleitkomma-Variablen, Ganzzahl-Variablen und String-Variablen (alphanumerische Zeichenketten).

Die Gleitkomma-Variablen können zunächst Werte zwischen -999999999 und +999999999 annehmen - mit neun Stellen Genauigkeit. Ist eine Zahl größer, als sich mit neun Stellen ausdrücken läßt, dann erfolgt die Zahlendarstellung in wissenschaftlicher Notation. Die Gleitkomma-Variable setzt sich dann aus Mantisse, Buchstabe E (für Exponent) und dem eigentlichen Zehner-Exponenten zusammen. Für Exponenten gilt der Bereich von -39 bis +38. Die Zahl 12345678901 wird dann z.B. so dargestellt: 1.23456789E+10.

Ganzzahl-Variablen (Integer) können für ganze Zahlen im Bereich von +32767 bis -32768 (ohne Komma oder Dezimalpunkt!) verwendet werden. Integer sind Zahlen wie 5, 10 oder auch -100. Integer-Zahlen verbrauchen in Feldern wesentlich weniger Speicherplatz als Gleitkomma-Variablen.

String-Variablen werden für Textstücke aus Buchstaben, Zahlen und anderen Zeichen benötigt. Ein typischer Inhalt einer String-Variablen ist z.B. "COMMODORE 16".

```
*********************
*  VARIABLEN-NAMEN  *
*********************
```

Variablen-Namen bestehen aus einem Buchstaben oder einem Buchstaben, gefolgt von einer Zahl, oder aus zwei Buchstaben. Variablen-Namen können auch länger sein, aber zur Erkennung für den Computer zählen nur die ersten zwei Zeichen. In einem Namen darf kein reservierter Name (wie ST oder TI) und kein BASIC-Schlüsselwort (wie ON oder IF) enthalten sein!

Die Integer-Variable (Ganzzahl-Variable) wird durch ein zusätzliches Prozentzeichen ('%') im Anhang an den Variablen-Namen gekennzeichnet. String- oder Text-Variable führen nach dem Namen das Dollarzeichen ('$').

BEISPIELE:

Gleitpunkt-Variablen-Namen:	A, A5, BZ
Integer-Variablen-Namen:	A%, A5%, BZ%
String-Variablen-Namen:	A$, A5$, BZ$

```
*********************
*  MATRIZEN/FELDER  *
*********************
```

Eine Matrix, auch Feld genannt, besteht aus einer Anzahl Variablen, die alle den gleichen Namen tragen, aber zur Festlegung ihrer Position im Matrixfeld eine zusätzliche Indexzahl haben. Die Größe der Felder wird mit der DIM-Anweisung festgelegt, und die Matrix selbst kann für Fließkomma-, Ganzzahl- oder String-Variable festgelegt werden. Dem Matrix-Variablen-Namen folgen - in Klammern - die Indexzahl(en). Der niedrigste benutzbare Indexwert ist immer 0.

BEISPIELE:

A (7),BZ%(11),A$(87),WW(0)

Solche (eindimensionale) Felder werden auch 'Vektoren' genannt. Eine Matrix kann jedoch mehr als eine Dimension haben. Ein zweidimensionales Feld besteht aus Zeilen und Spalten, wobei die erste Zahl nach der Variablen (in Klammern) die Zeile und die zweite Zahl (in Klammern) die Spalte definiert - wie beim Koordinatengitter einer Landkarte. Analog kann man sich bei dreidimensionalen Matrizen ein räumliches Gitter vorstellen, während bei höheren Dimensionen die grafische Anschauung versagt.

BEISPIELE:

A(7,2),BZ%(2,2,4),Z$(3,2),W7(0,5)

```
**********************************
*  RESERVIERTE VARIABLEN-NAMEN  *
**********************************
```

Der COMMODORE 16 kennt sieben Variablen-Namen, die reserviert sind, d.h. die für andere, als die vorgesehene Verwendung, nicht benützt werden können. Es sind dies die Variablen: DS, DS$, ER, EL, ST, TI und TI$. Außerdem können Schlüsselworte, wie TO und IF, nicht als freie Variablen-Namen eingesetzt werden. Es ist auch nicht zulässig, daß Variable gebildet werden, die Schlüsselworte enthalten, wie z.B. SRUN, RNEW oder XLOAD.

ST ist die Status-Variable für alle Ein- und Ausgabe-Operationen, mit Ausnahme normaler Bildschirm- bzw. Tastatur-Operationen. Der in ST stehende Wert ist vom Ergebnis der letzten I/O-Operation abhängig. Detailinformationen entnehmen Sie dem Programmierhandbuch. Eins jedoch vorab: Ist der Statuswert =0, dann verlief der I/O-Vorgang fehlerfrei.

TI und TI$ sind Variablen, die mit der im COMMODORE 16 eingebauten Echtzeituhr direkt gekoppelt sind. Die Systemuhr wird alle 60-tel Sekunden neu gesetzt. Beim Einschalten des COMMODORE 16 startet die Uhr bei 0. Mit jeder Neuzuweisung an die Variable TI$ verändert sich ihr Wert. Über die Variable TI kann Echtzeit ausgelesen werden.

TI$ liest den Wert der 24-Stunden Uhr als String. Die ersten zwei Ziffern stellen die Stunden, die 3. und 4. Ziffer die Minuten und die 5. und 6. Ziffer die Sekunden dar. Mit der Zuweisung TI$="hhmmss" kann jederzeit die Uhrzeit aktualisiert werden (hh=Stunden, mm=Minuten, ss=Sekunden). Die 6-stellige Angabe ist obligatorisch.

BEISPIEL:

TI$ = "142030" setzt die Uhr auf 14 Uhr 20 Minuten und 30 Sekunden.

Mit Zugriff auf die Variable DS kann der Befehlskanal des Disketten-Laufwerks gelesen und die Fehleranzeige des Laufwerks rückgesetzt werden (z.B. nach einer Fehlermeldung). Als Textstring wird die Variable DS$ abgefragt. Diese Statusabfragen sollten nach jeder Disketten-Operation wie z.B. DLOAD oder DSAVE erfolgen.

ER, EL sind Variablen, die in Fehlersuchroutinen (ERROR-TRAPPING) Verwendung finden. Mit ER kann der letzte Fehler seit dem Programmstart ausgelesen werden. EL meldet die Zeile, in der der Fehler auftrat. Mit ERR$ kann vom Programm aus eine der BASIC-Fehlermeldungen ausgedruckt werden. Mit PRINT ERR$(ER) wird die letzte Fehlermeldung ausgedruckt.

```
**********************
*  BASIC-OPERATOREN  *
**********************
```

Die Arithmetischen Operatoren schließen folgende Zeichen ein:

- `+` Addition
- `-` Subtraktion
- `*` Multiplikation
- `/` Division
- `↑` Potenzieren (Exponent)

Enthält eine Zeile mehrere Operatoren, dann tritt eine Prioritätenfolge in Kraft, d.h. bestimmte Operationen werden vor anderen durchgeführt:

1. Potenzieren
2. Multiplikation und Division
3. Addition und Subtraktion

Treffen Operationen gleicher Priorität aufeinander, dann erfolgt der Ablauf der Operation von links nach rechts. Wird eine anderslautende Ablaufordnung gewünscht, dann kann dies im COMMODORE 16 BASIC durch Klammern erreicht werden. Operationen in Klammern werden vorrangig ausgeführt. Dabei ist darauf zu achten, daß alle geöffneten Klammern wieder geschlossen werden, sonst kommt es bei laufendem Programm zum Abbruch mit der Fehlermeldung SYNTAX ERROR.

Zum Vergleich der Werte zweier Operanden, etwa zum Test auf Gleichheit oder Ungleichheit, stehen VERGLEICHSOPERATOREN zur Verfügung. Das Ergebnis ist -1 für 'wahr' und 0 für 'falsch'. Arithmetische Operatoren haben stets Vorrang vor Relationalen (Vergleichs-) Operatoren. Es gibt folgende Vergleichsoperatoren:

=	Gleich
<	Kleiner als
>	Größer als
<= oder =<	Kleiner oder gleich
>= oder =>	Größer oder gleich
<> oder ><	Ungleich

Abschließend gibt es drei LOGISCHE OPERATOREN, mit Priorität hinter den Arithmetischen wie den Relationalen Operatoren. Es sind dies:

AND	Logisches UND, aber auch bitweise Verknüpfung
OR	Logisches ODER, aber auch bitweise Verknüpfung
NOT	Logisches NICHT, aber auch bitweise Inversion + 1

Sie werden meist in Erweiterung von IF...THEN-Anweisungen genützt. Werden sie in Verbindung mit Arithmetischen Operatoren benützt, dann rangieren sie noch hinter + und -.

BEISPIELE:

IF A=B AND C=D THEN 100	Erfordert beide Teilbedingungen A=B, C=D.
IF A=B OR C=D THEN 100	Erfordert mindestens eine Teilbedingung.
A=5:B=4:PRINT A=B	Druckt einen Wert von 0 aus.
A=5:B=4:PRINT A>B	Druckt einen Wert von -1 aus.
PRINT 123 AND 15:PRINT 5 OR 7	Druckt Werte von 11 und 7 aus.

```
********************************
*                              *
*    A  N  H  A  N  G          *
*                              *
********************************
```


* BEFEHLS-ABKÜRZUNGEN
* FEHLERMELDUNGEN
* DISK-FEHLERMELDUNGEN
* ABGELEITETE MATHEMATISCHE FUNKTIONEN
* NOTENWERTE
* KODE-TABELLEN
* MASCHINEN-MONITOR TEDMON
* BITS UND BYTES
* SPEICHERBELEGUNG
* INDEX

```
**************************
*  BEFEHLS - ABKÜRZUNGEN  *
**************************
```

Allgemeines Prinzip: wenn ein Schlüsselwort nicht ausgeschrieben wird, muß der zuletzt getippte Buchstabe mit <Shift> eingegeben werden. Zur Verdeutlichung ist die Schreibweise des Text-Modus gewählt. Der Typ des Befehls ist in der Tabelle wie folgt gekennzeichnet:

A = Anweisung, K = Kommando, O = Operator
NF = numerische Funktion, SF = String-Funktion
NR = reservierte numerische Variable, SR = reserv. String-Variable

Schlüsselwort	Abkürzung	Typ	Schlüsselwort	Abkürzung	Typ
abs	aB	NF	fn	-	NF
and	aN	O	for	fO	A
asc	aS	NF	fre	fR	NF
atn	aT	NF	get	gE	A
auto	aU	K	getkey	getkE	A
backup	bA	K	get#	gE#	A
box	bO	A	gosub	goS	A
char	chA	A	goto	gO	A
chr$	cH	SF	graphic	gR	A
circle	cI	A	gshape	gS	A
close	clO	A	header	heA	K
clr	cL	A	hex$	hE	SF
cmd	cM	A	if	-	A
collect	colL	K	input	-	A
color	coL	A	input#	iN	A
cont	cO	K	instr	inS	NF
copy	coP	K	int	-	NF
cos	-	NF	joy	jO	NF
data	dA	A	key	kE	K
dec	-	NF	left$	leF	SF
def	dE	A	len	-	NF
delete	deL	K	let	lE	A
dim	dI	A	list	lI	K
directory	diR	K	load	lO	K
dload	dL	K	locate	loC	A
do	-	A	log	-	NF
draw	dR	A	loop	loO	A
ds	-	NR	mid$	mI	SF
ds$	-	SR	monitor	mO	A
dsave	dS	K	new	-	K
el	-	NR	next	nE	A
end	eN	A	not	nO	O
er	-	NR	on	-	A
err$	eR	SF	open	oP	A
exp	eX	NF	or	-	O

Schlüsselwort	Abkürzung	Typ	Schlüsselwort	Abkürzung	Typ
paint	pA	A	save	sA	K
peek	pE	NF	scale	scA	A
poke	pO	A	scnclr	sC	A
pos	-	NF	scratch	scR	K
print	?	A	sgn	sG	NF
print#	pR	A	sin	sI	NF
printusing	?usI	A	sound	sO	A
pudef	pU	A	spc(	sP	nach PRINT
rclr	rC	NF	sqr	sQ	NF
rdot	rD	NF	sshape	sS	A
read	rE	A	st	-	NR
rem	-	A	stop	sT	A
rename	reN	K	str$	stR	SF
renumber	renU	K	sys	sY	A
restore	reS	A	tab(	tA	nach PRINT
resume	resU	A	tan	-	NF
return	reT	A	ti	-	NR
rgr	rG	NF	ti$	-	SR
right$	rI	SF	trap	tR	A
rlum	rL	NF	troff	troF	A
rnd	rN	NF	tron	trO	A
run	rU	K	until	uN	A
			usr	uS	NF
			val	vA	NF
			verify	vE	K
			vol	vO	A
			wait	wA	A
			while	wH	A

```
********************
*  FEHLERMELDUNGEN  *
********************
```

NR.	FEHLERMELDUNG	BESCHREIBUNG
1	TOO MANY FILES	Es dürfen höchstens 10 Dateien gleichzeitig geöffnet werden.
2	FILE OPEN	Es wurde versucht, eine bereits geöffnete Datei zu öffnen.
3	FILE NOT OPEN	Die in einer I/O-Anweisung angegebene Datei muß vorher geöffnet worden sein.
4	FILE NOT FOUND	Diskette: die gesuchte Datei existiert nicht, Kassette: Bandende-Markierung (EOT) gelesen.
5	DEVICE NOT PRESENT	Das Peripheriegerät ist nicht ansprechbar.
6	NOT INPUT FILE	Es wurde mit GET# oder INPUT# versucht, aus einer zum Schreiben geöffneten Datei zu lesen.
7	NOT OUTPUT FILE	Es wurde mit PRINT# versucht, in eine zum Lesen geöffnete Datei zu schreiben.
8	MISSING FILE NAME	Ein OPEN-, LOAD- oder SAVE-Befehl braucht immer einen Dateinamen.
9	ILLEGAL DEVICE NUMBER	Es wurde versucht, ein Gerät anzusprechen, das nicht paßt (z.B. SAVE auf Bildschirm).
10	NEXT WITHOUT FOR	Entweder sind Schleifen nicht korrekt verschachtelt oder nach NEXT steht ein nicht passender Variablenname.
11	SYNTAX	Ein BASIC-Befehl ist nicht korrekt geschrieben. Es kann sich um falsch geschriebene Schlüsselwörter, fehlende Klammern oder überzählige Zeichen handeln.
12	RETURN WITHOUT GOSUB	Eine RETURN-Anweisung wurde erreicht, obwohl vorher kein GOSUB-Befehl gegeben wurde.
13	OUT OF DATA	Eine READ-Anweisung wurde erreicht, ohne daß noch Daten in DATA-Zeilen vorhanden sind, die nicht bereits abgearbeitet wurden.
14	ILLEGAL QUANTITY	Eine Funktion wurde mit einem außerhalb des erlaubten Zahlenbereichs liegenden Argument aufgerufen.

NR.	FEHLERMELDUNG	BESCHREIBUNG
15	OVERFLOW	Das Ergebnis einer Berechnung ist betragsmäßig größer als die höchste zulässige Zahl (1.701411833E+38).
16	OUT OF MEMORY	Entweder ist kein Speicherplatz für Programme und Variablen vorhanden, oder es sind zu viele FN-, DO-, FOR- oder GOSUB-Anweisungen aktiv.
17	UNDEF'D STATEMENT	Eine angesprochene Zeilennummer existiert nicht im Programm.
18	BAD SUBSCRIPT	Ein Feldindex außerhalb des in der DIM-Anweisung festgelegten Bereichs wurde angesprochen.
19	REDIM'D ARRAY	Ein schon eingerichtetes Feld kann nicht ein zweitesmal DIMensioniert werden. Wenn ein Feld mit einem Index zwischen 0 und 10 angesprochen wird, bevor es DIMensioniert wurde, wird es automatisch mit 10 DIMensioniert.
20	DIVISION BY ZERO	Es darf nicht durch Null geteilt werden.
21	ILLEGAL DIRECT	INPUT- oder GET-Anweisungen dürfen nur im Programm-Modus verwendet werden.
22	TYPE MISMATCH	Anstelle einer Zeichenkette darf kein numerischer Ausdruck verwendet werden und umgekehrt.
23	STRING TOO LONG	Eine Zeichenkette darf nur bis zu 255 Zeichen enthalten. Programmzeilen und INPUT-Antworten dürfen nur 80 Zeichen enthalten.
24	FILE DATA	Entspricht TYPE MISMATCH bei Eingabe von Kassette oder Diskette.

```
****************************
*  DISK - FEHLERMELDUNGEN  *
****************************
```

Diese Fehlermeldungen werden durch die reservierten Variablen DS (Fehler-Nr.) und DS$ (String aus: Fehler-Nr., Fehlermeldungstext, Spur, Sektor [, Drive-Nr.]) ausgegeben.

Fehlermeldungen mit Nummern unter 20 brauchen nicht berücksichtigt zu werden, mit Ausnahme von 01, die in der Spur-Nr. innerhalb DS$ angibt, wieviele Files durch einen SCRATCH-Befehl gelöscht wurden.

Das DOS ist das Disk Operating System (Disketten-Betriebssystem), das im Laufwerk über einen eigenen RAM-Speicher verfügt.

NR.	FEHLERMELDUNG	BESCHREIBUNG
20	READ ERROR	Blockheader nicht gefunden. Der Disk-Controller ist nicht in der Lage, den Header des benötigten Datenblocks zu finden. Wird hervorgerufen durch eine unzulässige Sektornummer oder durch Zerstörung des Headers.
21	READ ERROR	Kein Synchronisierungs-Zeichen. Der Disk-Controller ist nicht in der Lage, ein Synchronzeichen auf der verlangten Spur zu finden. Wird verursacht durch Dejustage des Schreib-/Lesekopfs, keine oder unformatierte Diskette im Laufwerk oder falsch eingelegte Diskette. Kann auch Hardware-Fehler anzeigen.
22	READ ERROR	Datenblock nicht vorhanden. Der Disk-Controller sollte einen Datenblock lesen oder prüfen, der nicht korrekt geschrieben wurde. Diese Fehlermeldung tritt in Verbindung mit Direktzugriffs-Befehlen auf und zeigt an, daß eine unzulässige Spur- und/oder Sektornummer angegeben wurde.
23	READ ERROR	Prüfsumme fehlerhaft im Datenblock. Dieser Fehler zeigt an, daß in einem oder mehreren Datenbytes ein Fehler vorliegt. Die Daten wurden in den DOS-Bereich geladen, aber die Prüfsumme ist falsch. Diese Meldung kann auch an schlechter Erdung liegen.
24	READ ERROR	Byte-Entschlüsselungsfehler. Die Daten wurden in den DOS-Bereich gelesen, aber aufgrund eines unzulässigen Bit-Musters im Datenbyte entstand ein Hardware-Fehler. Diese Meldung kann auch an schlechter Erdung liegen.
25	WRITE ERROR	Fehler bei der Überprüfung des Geschriebenen. Diese Meldung entsteht, wenn der Controller eine Abweichung zwischen den geschriebenen Daten und jenen im DOS-Bereich entdeckt.

NR.	FEHLERMELDUNG	BESCHREIBUNG
26	WRITE PROTECT ON	Es wurde versucht, auf eine Diskette mit aufgeklebtem Schreibschutz zu schreiben.
27	READ ERROR	Prüfsumme fehlerhaft im Header. Der Disk-Controller hat einen Fehler im Header des angesprochenen Datenblocks gefunden. Der Block wurde nicht in den DOS-Bereich eingelesen. Diese Melkann auch an schlechter Erdung liegen.
28	WRITE ERROR	Zu langer Datenblock. Der Controller versucht, das Synchronzeichen des nächsten Headers zu finden, nachdem ein Datenblock geschrieben wurde. Wenn das Synchronzeichen nicht innerhalb einer bestimmten Zeit auftaucht, entsteht die Fehlermeldung. Der Fehler wird durch schlecht formatierte Disketten (die Daten reichen bis in den nächsten Block) oder Hardwarefehler verursacht.
29	DISK ID MISMATCH	Diese Meldung tritt auf, wenn der Controller auf einen Datenblock zugreifen soll, dessen ID nicht mit der im Directory übereinstimmt. Die Meldung kann durch einen zerstörten Header entstehen.
30	SYNTAX ERROR	Das DOS kann den Befehl im Befehlskanal nicht verstehen (unzulässige Anzahl oder Länge von Dateinamen, falsches Muster z.B. im COPY-Befehl, usw.).
31	SYNTAX ERROR	Unzulässiger Befehl. Der Befehl muß an erster Stelle einer Befehlszeile stehen.
32	SYNTAX ERROR	Der Befehl ist länger als 58 Zeichen.
33	SYNTAX ERROR	Unzulässiger Dateiname, z.B. falsche Verwendung des 'Jokers' im OPEN- oder SAVE-Befehl.
34	SYNTAX ERROR	In einem Befehl wurde der Dateiname ausgelassen oder vom DOS nicht als solcher erkannt. Häufig wurde ein Doppelpunkt (:) ausgelassen.
39	SYNTAX ERROR	Dieser Fehler kann auftreten, wenn der über den Befehlskanal (Sekundäradresse 15) geschickte Befehl vom DOS nicht erkannt wird.

NR.	FEHLERMELDUNG	BESCHREIBUNG
50	RECORD NOT PRESENT	Tritt auf, wenn auf einen Record einer relativen Datei zugegriffen wird, der noch nicht angelegt ist (Record-Nummer zu hoch). Kann ignoriert werden, wenn die Datei mit einem PRINT#-Befehl erweitert werden soll. GET# oder INPUT# dürfen nach dieser Meldung nicht ausgeführt werden!
51	OVERFLOW IN RECORD	Eine PRINT#-Anweisung überschreitet die Record-Grenze. Die Informationen werden abgeschnitten. Der Wagenrücklauf, der eine Eingabe abschließt und das Record-Ende markiert, muß bei der ausnutzbaren Länge berücksichtigt werden.
52	FILE TOO LARGE	Die für ein relatives File angegebene Record-Nummer ist zu groß und würde die Kapazität der Diskette überschreiten.
60	WRITE FILE OPEN	Diese Meldung entsteht, wenn eine nicht korrekt abgeschlossene Datei (kein CLOSE nach Beschreiben) wieder geöffnet werden soll.
61	FILE NOT OPEN	Es wurde versucht, eine nicht vorher geöffnete Datei zu benutzen. Manchmal wird die Meldung nicht ausgegeben und der Befehl ignoriert.
62	FILE NOT FOUND	Die gesuchte Datei ist nicht vorhanden.
63	FILE EXISTS	Es wurde versucht, ohne Überschreibungs-Kennzeichnung auf eine schon bestehende Datei zu schreiben.
64	FILE TYPE MISMATCH	Der Typ der angesprochenen Datei stimmt nicht mit dem im Directory vermerkten Typ überein.
65	NO BLOCK	Ein Block, der mit dem B-A-Befehl als belegt gekennzeichnet werden sollte, ist bereits belegt. Die Spur- und Sektornummer in DS$ zeigen den nächsten freien Block an; sind beide Werte Null, sind alle höhernumerierten Blocks belegt.

NR.	FEHLERMELDUNG	BESCHREIBUNG
66	ILLEGAL TRACK AND SECTOR	Das DOS hat versucht, eine Spur oder einen Sektor anzusprechen, die/der im gegebenen Format nicht vorhanden ist. Das kann von Problemen beim Lesen des Zeigers zum nächsten Block herrühren.
67	ILLEGAL SYSTEM T OR S	Diese besondere Fehlermeldung weist auf eine unzulässige Systemspur oder einen unzulässigen Systemsektor hin.
70	NO CHANNEL	Im RAM der Floppy sind alle Kanäle (interne Pufferspeicher) belegt. Je nach Floppy kann nur eine bestimmte Anzahl von Files (abhängig auch von deren Typ) gleichzeitig geöffnet sein.
71	DIRECTORY ERROR	Die BAM (Block Availability Map, Blockverfügbarkeitstabelle) kann nicht gelesen werden. Eventuell wurde sie im DOS-Speicher überschrieben. Um dieses Problem zu beseitigen, initialisieren Sie die Diskette, um die BAM wieder in den DOS-Speicher einzulesen. Vorher geöffnete Dateien können dadurch verstümmelt werden.
72	DISK FULL	Entweder sind alle Blocks der Diskette belegt, oder die Maximalzahl der Directory-Einträge ist erreicht. Die Meldung kommt, wenn z.B. auf der 1541 noch zwei Blocks verfügbar sind, um der momentanen Datei zu ermöglichen, korrekt abgeschlossen zu werden.
73	DOS MISMATCH	(73, CBM DOS V 2.6 1541) DOS 1 und 2 sind lese- aber nicht schreib-kompatibel. Disketten können zwar auf einem Laufwerk des anderen Formats gelesen werden. Bei einem Schreibversuch kommt jedoch diese Fehlermeldung. (Es gibt aber Hilfsprogramme, die ein Format in das andere umwandeln.) Die Meldung mit der DOS-Versionsnummer kann immer direkt nach dem Einschalten über den Fehlerkanal abgefragt werden.
74	DRIVE NOT READY	Es wurde versucht, ein Floppy-Laufwerk, das keine oder eine unformatierte Diskette oder eine mit einem total inkompatiblem Format enthält, anzusprechen.

```
*****************************************
* ABGELEITETE MATHEMATISCHE FUNKTIONEN *
*****************************************
```

Einige Funktionen, die in BASIC 3.5 nicht vordefiniert sind, können mit Hilfe der folgenden Formeln berechnet werden:

```
SEKANS                          SEC(X)   =1/COS(X)
COSEKANS                        CSC(X)   =1/SIN(X)
COTANGENS                       COT(X)   =1/TAN(X)

ARCUSSINUS                      ARCSIN(X)=ATN(X/SQR(1-X*X))
ARCUSCOSINUS                    ARCCOS(X)= π /2 - ATN(X/SQR(1-X*X))
ARCUSCOTANGENS                  ARCCOT(X)= π /2 - ATN(X)

ARCUSSEKANS                     ARCSEC(X)= π /2 - ATN(1/SQR(X*X-1))
ARCUSCOSEKANS                   ARCCSC(X)=ATN(1/SQR(X*X-1))

SINUS HYPERBOLICUS              SINH(X)  =(EXP(X)-EXP(-X))/2
COSINUS HYPERBOLICUS            COSH(X)  =(EXP(X)+EXP(-X))/2
TANGENS HYPERBOLICUS            TANH(X)  =1-2*EXP(-X)/(EXP(X)+EXP(-X))
COTANGENS HYPERBOLICUS          COTH(X)  =1+2*EXP(-X)/(EXP(X)-EXP(-X))

SEKANS HYPERBOLICUS             SECH(X)  =2/(EXP(X)+EXP(-X))
COSEKANS HYPERBOLICUS           CSCH(X)  =2/(EXP(X)-EXP(-X))

AREASINUS HYPERBOLICUS          ARSINH(X)=LOG(X+SQR(X*X+1))
AREACOSINUS HYPERBOLICUS        ARCOSH(X)=LOG(X+SQR(X*X-1))
AREATANGENS HYPERBOLICUS        ARTANH(X)=LOG((1+X)/(1-X))/2
AREACOTANGENS HYPERBOLICUS      ARCOTH(X)=LOG((X+1)/(X-1))/2

AREASEKANS HYPERBOLICUS         ARSECH(X)=LOG((1+SQR(1+X*X))/X)
AREACOSEKANS HYPERBOLICUS       ARCSCH(X)=LOG((1+SQR(1-X*X))/X)

BRIGGS-LOG  (Basis 10)          LG(X)    =LOG(X)/LOG(10)


Runden ganzer Zahlen:           DEFFNI(X)=INT(X+.5)
Runden auf ganze Hundertstel:   DEFFNP(X)=INT(X*100+.5)/100
```

```
****************
*  NOTENWERTE  *
****************
```

NOTE	SOUNDREGISTER-WERT	FREQUENZ (Hz)
A	7	110
H	118	123.5
C	169	130.8
D	262	146.8
E	345	164.7
F	383	174.5
G	453	195.9
A	516	220.2
H	571	246.9
C	596	261.4
D	643	293.6
E	685	330
F	704	349.6
G	739	392.5
A	770	440.4
H	798	494.9
C	810	522.7
D	834	588.7
E	854	658
F	864	699
G	881	782.2
A	897	880.7
H	911	989.9
C	917	1045
D	929	1177
E	939	1316
F	944	1398
G	953	1575

Die obige Tabelle enthält die Soundregisterwerte für die Noten von vier Oktaven. Die Werte werden als zweiter Parameter beim SOUND-Befehl benötigt. Um die erste Note in der Tabelle (A mit Soundregisterwert 7) erklingen zu lassen, geben Sie den Befehl SOUND 1,7,30 ein. Mit den folgenden Formeln können Sie die Soundregisterwerte für nicht in der Tabelle enthaltene Frequenzen bestimmen, die noch vom verwendeten Farbfernsehsystem abhängen (in Deutschland PAL, die Tabellenwerte stimmen unabhängig davon gut überein):

SOUNDREGISTERWERT = 1024 - (111860.781 / Frequenz[Hz]) (NTSC)
SOUNDREGISTERWERT = 1024 - (111840.45 / Frequenz[Hz]) (PAL)

****
* KODE - TABELLEN *
****

BILDSCHIRM - KODE
****

Die folgende Tabelle listet alle Zeichen auf, über die der COMMODORE 16 auf dem Bildschirm verfügt. Sie zeigt, welche Zahlen in den Bildschirmspeicher (3072 bis 4095) gePOKEt werden müssen, um ein gewünschtes Zeichen zu erhalten. Die Tabelle zeigt auch, welches Zeichen einer Zahl entspricht, die vom Bildschirm gePEEKt wurde.

Von den zwei Zeichensätzen ist jeweils nur einer zur Zeit benutzbar. Das bedeutet, Sie können nicht gleichzeitig Zeichen beider Sätze auf dem Bildschirm haben. Die Sätze werden durch gleichzeitiges Niederhalten der Tasten <Shift> und <C=> gewechselt. Im BASIC schaltet PRINT CHR$(142) in den Großbuchstaben-/Grafik-Modus und PRINT CHR$(14) in den Groß-/Kleinbuchstaben-Modus.

Jedes Zeichen in der Tabelle kann auch REVERS angezeigt werden. Den Kode des reversen Zeichen erhalten Sie, wenn Sie zum angegebenen Wert 128 addieren.

Satz 1	Satz 2	POKE	Satz 1	Satz 2	POKE	Satz 1	Satz 2	POKE
@		0	T	t	20	(		40
A	a	1	U	u	21	)		41
B	b	2	V	v	22	*		42
C	c	3	W	w	23	+		43
D	d	4	X	x	24	,		44
E	e	5	Y	y	25	–		45
F	f	6	Z	z	26	.		46
G	g	7	[		27	/		47
H	h	8	£		28	0		48
I	i	9	]		29	1		49
J	j	10	↑		30	2		50
K	k	11	←		31	3		51
L	l	12	**SPACE**		32	4		52
M	m	13	!		33	5		53
N	n	14	"		34	6		54
O	o	15	#		35	7		55
P	p	16	$		36	8		56
Q	q	17	%		37	9		57
R	r	18	&		38	:		58
S	s	19	'		39	;		59

Satz 1	Satz 2	POKE	Satz 1	Satz 2	POKE	Satz 1	Satz 2	POKE
<		60		T	84			108
=		61		U	85			109
>		62		V	86			110
?		63		W	87			111
		64		X	88			112
	A	65		Y	89			113
	B	66		Z	90			114
	C	67			91			115
	D	68			92			116
	E	69			93			117
	F	70			94			118
	G	71			95			119
	H	72	SPACE		96			120
	I	73			97			121
	J	74			98		✓	122
	K	75			99			123
	L	76			100			124
	M	77			101			125
	N	78			102			126
	O	79			103			127
	P	80			104			
	Q	81			105			
	R	82			106			
	S	83			107			

Die Kodes 128 bis 255 bilden die reversen Zeichen zu den Kodes 0 bis 127.

ASC- UND CHR$-KODES
****

Diese Tabelle zeigt Ihnen alle möglichen Zeichen (inclusive Steuerzeichen), die abhängig von X erscheinen, wenn Sie PRINT CHR$(X) eingeben. Ebenfalls angegeben werden die Werte, die Sie erhalten, wenn Sie PRINT ASC("A") eingeben, wobei A jedes eintippbare Zeichen (auch Steuerzeichen) sein kann. Dies ist nützlich, wenn Sie das in einer GET-Anweisung erhaltene Zeichen auswerten, Groß-/Kleinbuchstaben-Wandlung durchführen oder auf Zeichen basierende Befehle (wie Schalten zum Groß-/Kleinbuchstaben-Modus), die nicht in Anführungszeichen geschlossen werden können, drucken wollen.

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
	0	↓	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
WHT	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
DISABLES SHIFT C=	8		25	*	42	;	59
ENABLES SHIFT C=	9		26	+	43	<	60
	10	ESCAPE	27	,	44	=	61
	11	RED	28	–	45	>	62
	12	→	29	.	46	?	63
RETURN	13	GRN	30	/	47	@	64
SWITCH TO LOWER CASE	14	BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
D	68	♠	97		126		155
E	69		98		127	PUR	156
F	70		99		128	←	157
G	71		100	♠	129	YEL	158
H	72		101	FLASH ON	130	CYN	159
I	73		102	FLASH OFF	131	SPACE	160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	HELP	140		169
S	83		112	SHIFT RETURN	141		170
T	84	●	113	SWITCH TO UPPER CASE	142		171
U	85		114		143		172
V	86	♥	115	BLK	144		173
W	87		116	↑	145		174
X	88		117	RVS OFF	146		175
Y	89	╳	118	CLR HOME	147		176
Z	90	○	119	INST DEL	148		177
[	91	♣	120		149		178
£	92		121	╳	150		179
]	93	♦	122	○	151		180
↑	94		123	♣	152		181
←	95		124		153		182
	96		125	♦	154		183

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
	184		186		188		190
	185		187		189		191

Kodes 192-223 identisch mit 96-127
Kodes 224-254 identisch mit 160-190
Kode 255 identisch mit 126

Bildschirm - Steuerzeichen
****

Bei der Eingabe einer in Anführungszeichen eingeschlossenen Zeichenkette (String) von der Tastatur befindet sich der Rechner im 'Anführungszeichen-Modus', der jeweils nach dem Eintippen weiterer Anführungszeichen aus- und eingeschaltet wird. In dieser speziellen Betriebsart werden die Bildschirm-Steuerbefehle von der Tastatur (z.B. Cursor Home) nicht ausgeführt sondern als Steuerzeichen in die Zeichenkette eingefügt und gelangen erst beim Abruf des Strings durch einen PRINT-Befehl zur Ausführung. Im Programmlisting werden diese Steuerzeichen durch revers dargestellte Platzhalter-Symbole sichtbar gemacht. Sie finden diese Symbole in der ASC- und CHR$-Tabelle, wenn Sie die Kodezahl des Steuerzeichens (Cursor abwärts z.B. 17) nehmen, 64 aufaddieren (hier 81) und das dazugehörige Zeichen invers darstellen (hier im Groß-/Grafik-Modus ein großes 'Q', im Groß-/Kleinbuchstaben-Modus ein kleines 'q'). Man findet diese Symbole häufig in Programmlistings, die mit einem Matrix-Drucker erstellt wurden. Beim Eintippen eines solchen Programms ist also jeweils die dem Platzhalter-Symbol entsprechende Steuertaste (hier Cursor ab) zu betätigen.

MASCHINENSPRACHEN-MONITOR TEDMON
****

Einführung

TEDMON ist ein eingebautes Maschinensprachenprogramm, mit dem Sie leicht Programme in Maschinensprache schreiben und testen können. TEDMON enthält einen Monitor für Maschinensprache, einen Mini-Assembler und einen Disassembler.

Programme in Maschinensprache, die mit TEDMON geschrieben werden, können eigenständig ablaufen, oder von BASIC aus als Unterprogramm aufgerufen werden. Dabei muß der letzte Befehl des Unterprogramms ein RTS (Return from Subroutine, $60) sein.

TEDMON-Befehlssatz:

A	Assemble	Wandelt ein Mnemonik (Klartext-Befehl) in den entsprechenden Maschinenkode des 6502 bzw. 7501.
C	Compare	Vergleicht zwei Speicherbereiche und zeigt die Unterschiede.
D	Disassemble	Wandelt Maschinenkode in Mnemoniks (Klartext).
F	Fill	Füllt einen Speicherbereich mit wählbarem Wert.
G	Go	Startet Maschinenprogramm an angegebener Adresse.
H	Hunt	Durchsucht einen Speicherbereich nach einem bestimmten Wert und zeigt alle Speicherplätze an, die diesen Wert enthalten.
L	Load	Lädt ein Programm von Kassette oder Diskette.
M	Memory	Zeigt alle Inhalte eines wählbaren Speicherbereichs in Hexdarstellung an.
R	Registers	Ausgabe der aktuellen Registerinhalte.
S	Save	Speichert ein Programm auf Kassette oder Diskette.
T	Transfer	Blockkopierbefehl, kopiert einen bestimmten Speicherbereich in einen anderen.
V	Verify	Vergleicht ein Programm im Arbeitsspeicher mit einem auf Kassette oder Diskette.
X	Exit	Zurück zu BASIC.
.	(Punkt)	Entspricht dem A (Assemble).
>	(größer als)	Ändert bis zu 8 Bytes ab bestimmter Speicherstelle (nach M-Befehl).
;	(Semikolon)	Ändert die 7501-Registerinhalte (nach R-Befehl).

Mit der Speicherstelle $07F8 ist es möglich, im Speicherbereich ab $8000 zwischen ROM (BASIC, Kernal) und RAM zu wählen. Hat die Speicherstelle den Wert 00, kann das ROM gelesen werden, beim Wert $80 das darunterliegende RAM. Das ist oft nützlich bei der Entwicklung von Maschinenprogrammen. Beachten Sie, daß die Speicherstelle $07F8 keinen Einfluß auf den Go-Befehl hat, dieser startet ein Programm abhängig von der aktuellen Speicherkonfiguration (RAM oder ROM aktiv) im jeweiligen Bereich, unabhängig vom Inhalt in $07F8. Beim COMMODORE 16 wird das nur 16 KByte große RAM dreimal in höhere Adreßbereiche 'gespiegelt', ein Zugriff per TEDMON auf z.B. RAM-Zelle $A000 wirkt daher auf $2000.

Benutzen von TEDMON:

Rufen Sie TEDMON auf, indem Sie eingeben:

MONITOR (oder die Abkürzung mO).

TEDMON antwortet, indem die Inhalte der Prozessor-Register angezeigt werden und der Cursor blinkt. Der Cursor ist Ihr 'Prompt', der Ihnen anzeigt, daß TEDMON auf Ihre Befehle wartet.

Beschreibung der einzelnen Befehle:

Befehl: A
Zweck: Eingeben einer Zeile Assemblerkode
Schreibweise: A <Adresse> <Mnemonischer Befehlskode> [<Operand>]

<Adresse> Eine Hexadezimalzahl, die die Speicherstelle angibt, wohin der Assemblerbefehl plaziert werden soll.
<Mnemonischer Befehlskode> Ein Klartext-Assemblerbefehl (z.B. LDA).
<Operand> Ob und was für ein Operand angegeben wird, legt die Adressierungsart des Befehls fest (z.B. sind für Zero-Page-Adressierung Werte zwischen 00 und $FF einzugeben, bei absoluter Adressierung 0000 bis $FFFF).

Die fertige Befehlszeile wird mit der Taste <Return> übergeben. Wenn die Zeile Fehler enthält, wird ein Fragezeichen angezeigt, und der Cursor geht zur nächsten Zeile. Der Fehler kann auf dem Schirm korrigiert werden.

Nachdem eine Zeile korrekt erstellt ist, wird ein Prompt angezeigt, der die nächste Speicherstelle schon bereitstellt, sodaß A und diese Speicherstelle nicht mehr eingetippt zu werden brauchen.

Beispiel:
```
A 1200 LDX #$2A
A 1202
```

Anmerkung: Ein Punkt (.) ist dem A-Befehl gleichgestellt.
Beispiel: .2000 LDA #$23

Befehl:	C
Zweck:	Vergleichen von zwei Abschnitten des Arbeitsspeichers
Schreibweise:	C <Adresse 1> <Adresse 2> <Adresse 3>

<Adresse 1>	Hexadezimalzahl, die den Anfang des zu vergleichenden Abschnitts angibt.
<Adresse 2>	Hexadezimalzahl, die das Ende davon angibt.
<Adresse 3>	Hexadezimalzahl, die den Anfang des anderen zu vergleichenden Abschnitts angibt.

Wenn die beiden Abschnitte des Arbeitsspeichers den gleichen Inhalt haben, gibt TEDMON "RETURN" aus. Bemerkt er Unterschiede, werden die Adressen der abweichenden Bytes angezeigt.

Befehl:	D
Zweck:	Disassemblieren, Zurückübersetzen von Maschinensprache in mnemonische Assemblerbefehle und Operanden.
Schreibweise:	D [<Adresse 1>] [<Adresse 2>]

<Adresse 1>	Hexadezimalzahl, die die Startadresse für die Rückübersetzung angibt.
<Adresse 2>	Hexadezimalzahl, die die Endadresse angibt.

Das Ausgabeformat des D-Befehls unterscheidet sich nur wenig von dem des A-Befehls. Das erste Zeichen einer Rückübersetzung ist ein Punkt statt eines "A". Zusätzlich werden die Hexadezimalzahlen der Kodes aufgelistet.

Eine Disassemblerliste kann über den Bildschirmeditor geändert werden. Ändern Sie den mnemonischen Befehlskode oder den Operanden und drücken Sie <Return>. Der A-Befehl wird aufgerufen und der geänderte Befehl in den Speicher geschrieben.

In der Disassemblerliste kann geblättert werden. Das Drücken eines D's (ohne Adreßangaben) bewirkt, daß die nächste Seite auf dem Schirm erscheint.

Beispiel:

```
D 3000 3004
. 3000  A9 00    LDA #$00
. 3002  FF       ???
. 3003  D0 2B    BNE $3030
```

Befehl: F
Zweck: Füllen eines Speicherbereichs mit einem bestimmten Bytewert.
Schreibweise: F <Adresse 1> <Adresse 2> <Byte>

<Adresse 1> Erste Adresse, die mit dem <Byte> zu füllen ist.
<Adresse 2> Letzte Adresse dieses Speicherbereichs.
<Byte> Ein- oder zweistellige Hexadezimalzahl angeben.

Beispiel: F 0400 0518 EA
Füllt den Speicherbereich von $0400 bis $0518 mit dem Wert $EA (einer NOP-Anweisung).

Befehl: G
Zweck: Startet die Ausführung eines Maschinenprogramms an der angegebenen Adresse.
Schreibweise: G [<Adresse>]

<Adresse> Eine Adresse (bis zu 4stellige Hexadezimalzahl), die den im R-Befehl angezeigten PC-Inhalt (PC = Program Counter = Befehlsadreßzeiger) verändert. Die Ausführung erfolgt ab bisherigem oder neu eingegebenen PC.

Der Go-Befehl stellt den alten Zustand der Register (die durch den R-Befehl angezeigt werden können) wieder her und beginnt die Ausführung an der (ggf. neu spezifizierten) PC-Adresse. Bei Anwendung des Go-Befehls ist Vorsicht geboten, damit der Rechner nicht durch ein fehlerhaftes Programm 'abstürzt'. Um nach Ausführen eines Maschinenprogramms zum TEDMON zurückzukehren, benutzen Sie die BRK-Anweisung ($00), nicht den RTS-Befehl!

Beispiel: G 1400
Die Ausführung beginnt mit der Speicherstelle $1400.

Befehl:	H
Zweck:	Durchsuchen eines Speicherbereichs nach angegebenen Inhalten.
Schreibweise:	H <Adresse 1> <Adresse 2> <Daten>

<Adresse 1>	Anfangsadresse des zu durchsuchenden Speicherbereichs.
<Adresse 2>	Endadresse davon.
<Daten>	Die gesuchten Daten können hexadezimal oder als ASCII-Zeichenkette angegeben werden. Bei der Angabe in ASCII muß dem ersten Zeichen ein Apostroph vorausgehen, z.B. 'ZEICHENKETTE. Die Daten können aus einem oder mehreren Elementen bestehen. Bei mehreren Elementen und hexadezimaler Schreibweise muß zwischen jeder Zahl eine Leerstelle stehen.

Beispiele:

```
H C000 FFFF 'READ
```

Suche im Bereich $C000 bis $FFFF nach der Zeichenkette "READ".

```
H A000 A101 A9 FF 4C
```

Suche im Bereich $A000 bis $A101 nach den Daten $A9, $FF und $4C.

Befehl:	L
Zweck:	Laden eines Programmfiles von Kassette oder Diskette.
Schreibweise:	L <"Filename">,<Geräte-#>

<"Filename">	Ein beliebiger, zulässiger Dateiname in Anführungszeichen.
<Geräte-#>	Eine Hexadezimalzahl, die die Gerätenummer angibt, 1 = Datassette, 8 = Floppy (oder 9, A, usw.).

Der LOAD-Befehl bewirkt, daß eine Programmdatei in den Arbeitsspeicher geladen wird. Die Anfangsadresse ist immer am Anfang eines Programms abgespeichert. Der LOAD-Befehl lädt ein Programm stets an dieselbe Stelle, von der es abgespeichert wurde. Dies ist für die Arbeit mit Maschinenprogrammen sehr wichtig, da nur wenige Programme 'relokierbar' sind, d.h. an anderer Stelle im Speicher ohne Änderungen lauffähig sind. Die Datei wird bis zu ihrer EOF-Marke (EOF = End Of File = Dateiendemarkierung) in den Arbeitsspeicher geladen.

Beispiele:

```
L "SCHIRM",1
```

Liest ein Programm "SCHIRM" von Kassette.

```
L "0:BLINKER",8
```

Lädt das Programm "BLINKER" vom Diskettenlaufwerk 0.

Befehl: M
Zweck: Anzeige des Speicherinhalts in Hexadezimaldarstellung und ASCII-Zeichen als Speicherauszug ('Dump').
Schreibweise: M [<Adresse 1>] [<Adresse 2>]

<Adresse 1> Startadresse des Speicherauszugs, frei wählbar. Wird sie nicht angegeben, wird von der um 96 erhöhten zuletzt verwendeten Startadresse ausgegangen.
<Adresse 2> Endadresse des anzuzeigenden Speicherbereichs, frei wählbar. Wird sie nicht angegeben, so wird die aktuelle Speicheradresse +96 angenommen.

Der Speicherinhalt wird in folgendem Format angezeigt (Beispiel):

```
>A048  41 E7 00 AA AA 00 98 56 :A!.**..V
```

Der Speicherinhalt kann durch den Bildschirmeditor geändert werden. Bringen Sie den Cursor zu den zu ändernden Hex-Daten, geben Sie die Korrektur ein und drücken Sie <Return>. Falls eine defekte RAM-Zelle gefunden oder ein Versuch gemacht wird, eine ROM-Speicherstelle zu ändern, wird das Fehlerflag (?) angezeigt.

Der ASCII-Speicherauszug der Daten wird REVERS (um sich von den anderen Daten abzuheben) rechts von den hexadezimalen Daten angezeigt. Wenn ein Zeichen nicht druckbar ist, wird es als reverser Punkt (.) angezeigt.

Wie beim DISASSEMBLER-Befehl können Sie auch hier durch Eingabe von M allein und Drücken der Taste <Return> weiterblättern.

Befehl: > (Größer-als-Zeichen)
Zweck: 1 bis 8 Speicherstellen können auf einmal gesetzt werden.
Schreibweise: > <Adresse> <Datenbyte 1> [<Datenbytes 2...8>]

<Adresse> Erste Speicherstelle, die gesetzt werden soll.
<Datenbyte 1> Bis zu zweistellige Hex-Daten, die in die erste Speicherstelle gesetzt werden sollen.
<Datenbytes 2...8> Hex-Daten, die in die folgenden Speicherstellen abgelegt werden sollen.

Dieser Befehl wird automatisch ausgeführt, wenn nach dem M-Befehl mit dem Bildschirmeditor Speicherinhalte geändert werden.

Beispiele:

```
>2000 08
```

Setzt eine 08 in Speicherstelle $2000.

```
>3000 23 45 65
```

Setzt eine $23 in Speicherstelle $3000, eine $45 in $3001 und eine $65 in $3002.

Befehl:	R
Zweck:	Anzeige der Prozessor-Registerinhalte, und zwar: Programmzähler PC, Prozessorstatusregister SR, Akkumulator AC, X- und Y-Indexregister XR und YR und den Stackpointer SP.
Schreibweise:	R

Beispiel:

```
R
   PC  SR AC XR YR SP
; 3020 00 20 00 10 F8
```

Anmerkung: Das Semikolon (;) kann in gleicher Weise zur Änderung der Registerinhalte benutzt werden wie das ">" beim M-Befehl.

Befehl:	S
Zweck:	Abspeichern des Speicherinhalts als Programmfile auf Kassette oder Diskette.
Schreibweise:	S <"Filename">,<Geräte-#>,<Adresse 1>,<Adresse 2>
<"Filename">	Beliebiger zulässiger Dateiname, der in Anführungszeichen eingeschlossen sein muß.
<Geräte-#>	Die beiden möglichen Gerätetypen sind Kassette und Diskette. Für Kassette geben Sie hier 1 an, für das Diskettenlaufwerk normalerweise 8 (bei mehreren auch 9 oder A, usw., siehe Floppy-Handbuch).
<Adresse 1>	Anfangsadresse des abzuspeichernden Speicherbereichs.
<Adresse 2>	Endadresse + 1 davon. Alle Daten bis ausschließlich dieser Adresse werden abgespeichert.

Mit diesem Befehl wird eine Programmdatei erzeugt, die Byte für Byte den Speicherinhalt wiedergibt. Als erste zwei Bytes wird die Anfangsadresse <Adresse 1> in LO-HI-Reihenfolge abgespeichert (Ladeadresse). Das Programm kann mit dem L-Befehl wieder geladen werden.

Beispiel: S "SPIEL",8,0400,0BFF
Speichert den Inhalt von $0400 bis $0BFE unter dem Namen "SPIEL" als Programmfile auf die Diskette.

Befehl: T
Zweck: Kopieren von Speicherinhalten in einen anderen Bereich des Arbeitsspeichers.
Schreibweise: T <Adresse 1> <Adresse 2> <Adresse 3>

<Adresse 1> Anfangsadresse der zu verschiebenden Daten.
<Adresse 2> Endadresse davon.
<Adresse 3> Anfangsadresse der neuen Plazierung, ab der die Daten abgespeichert werden sollen.

Daten können von einem niedrigen in einen höheren Speicherbereich übertragen werden und umgekehrt. Überlappen sich Ausgangs- und Zielbereich, so ist allerdings nur Rückwärtsverschieben möglich. Notfalls ist der Umweg über einen ganz anderen Speicherbereich zu nehmen.

Beispiel: T 1400 1600 3400
Der Speicherbereich von $1400 bis einschließlich $1600 wird nach $3400 bis $3600 kopiert.

Befehl: V
Zweck: Vergleichen einer Programmdatei auf Kassette oder Diskette mit dem Inhalt des Arbeitsspeichers.
Schreibweise: V <"Filename">,<Geräte-#>

<"Filename"> Beliebiger zulässiger Dateiname, der in Anführungszeichen eingeschlossen sein muß.
<Geräte-#> Hexadezimalzahl, die angibt, von welchem Gerät die Datei gelesen werden soll: 1 für Kassette, 8 (oder 9 oder A, usw.) für Diskette.

Der Verify-Befehl vergleicht eine Programmdatei mit dem Inhalt des Arbeitsspeichers. Der COMMODORE 16 zeigt auf dem Bildschirm "VERIFYING" an. Wenn eine Abweichung gefunden wird, wird "ERROR" ausgegeben. Wenn kein Fehler gefunden wurde, erscheint einfach wieder der blinkende Cursor.

Beispiel: V "TESTPRG",8
Vergleicht das Disketten-Programm "TESTPRG" mit dem aktuellen Speicherinhalt.

Befehl: X
Zweck: Rückkehr zu BASIC.
Schreibweise: X

Bei Ausführung des X-Befehls wird der Stackpointer SP auf den Wert gesetzt, wie er im R-Befehl angezeigt wurde. Wenn dieser irgendwie verändert wurde, benutzen Sie nach der Rückkehr ins BASIC den BASIC-Befehl CLR, um alle Zeiger auf korrekte Werte zurückzustellen.

```
********************
*  BITS UND BYTES  *
********************
```

Der Prozessor im COMMODORE 16 heißt 7501, er ist ein Nachfolgetyp des 6502 und ein 8-Bit-Prozessor. Mit jeder Speicherstelle werden also vom Prozessor gleichzeitig 8 Bits angesprochen, ein 'Byte'.

Mit 8 Bits lassen sich Zahlenwerte von 0-255 darstellen, nämlich 2*2*2*2 * 2*2*2*2 = 2 hoch 8 verschiedene, weil jedes Bit unabhängig von den anderen die 2 Zustände 0 oder 1 annehmen kann. Dieser Zahlenbereich ist bei PEEK- und POKE-Befehlen zu beachten, die den Inhalt einzelner Speicherstellen auslesen bzw. neu setzen.

Ein 8-Bit-Prozessor kann maximal 65536 = 2 hoch 16 Speicherstellen verwalten. Der Adreßbereich umfaßt die Adressen 0 bis 65535 (bei PEEK, POKE und SYS zu beachten). Dieser Umfang kommt dadurch zustande, daß der Rechner zur Speicheradressierung jeweils 2 Bytes, also insgesamt 16 Bits, verwendet, die er ggf. nacheinander aus dem Programmspeicher holen muß (z.B. bei Adressierungsart 'absolut').

Z.B. bei Verwendung der USR-Funktion muß man diese beiden Bytes (die die Anfangsadresse der eigenen Maschinenroutine angeben) selbst auf 2 aufeinanderfolgende Speicherstellen ('Pointer') schreiben. Die beiden Adreßbytes werden unterschieden in das LO-Byte, das die unteren 8 Binärstellen der insgesamt ja 16stelligen Adresse aufnimmt, und das HI-Byte mit den oberen 8 Binärstellen. Dadurch hat das HI-Byte gegenüber dem LO-Byte eine 256fache Wertigkeit. Allgemein muß bei diesem Prozessortyp die sog. 'LO-HI-Reihenfolge' eingehalten werden. Dabei liegt das LO-Byte im Speicher direkt <u>vor</u> dem HI-Byte!

```
Zerlegung einer Adresse AD in LO- und HI-Byte:
HI = INT( AD / 256 ): LO = AD - 256 * HI                .
Ermittlung einer vollen Adresse AD aus LO- und HI-Byte:
AD = LO + 256 * HI                                      .
Setzen eines Pointers in P und P+1 auf den Wert AD:
HI=INT(AD/256):LO=AD-256*HI:POKE P,LO:POKE P+1,HI       .
Lesen eines Pointers aus P und P+1 in die Variable AD:
AD = PEEK( P ) + 256 * PEEK( P+1 )                      .
```

Wie man leicht erkennt, ist die dezimale Schreibweise für Binärzahlen sehr umständlich. Daher wurden die 'Hexadezimal'-Zahlen eingeführt. In der dezimalen Stellenschreibweise ist ja eine Stelle jeweils 10mal mehr wert als die rechts davon, weil eine Stelle 10 verschiedene Werte (0-9) annehmen kann. Im Hexadezimalsystem werden je 4 Bits (ein 'Nibble') zu einer Stelle, einer Hexadezimalziffer, zusammengefaßt. Durch 4 Bits können aber 16 verschiedene Zahlenwerte (0-15) dargestellt werden, d.h. die 10 herkömmlichen Dezimalziffern reichen nicht aus. Für die neuen Hex-Ziffern mit den Wertigkeiten 10-15 wurden daher die Buchstaben A-F ausgewählt. In der hexadezimalen Stellen-Schreibweise ist folgerichtig eine Stelle jeweils 16mal mehr wert als die rechts davon (16*16=256, 16*256=4096, 16*4096=65536). Hex-Zahlen werden zur Unterscheidung mit einem vorangestellten Dollarzeichen ($) gekennzeichnet, z.B. $0F = 15 (dez.!) oder $51 = 81 = 5*16+1.

Ein Byte umfaßt 8 Bits, also zwei Hexstellen, und kann Werte von $00 bis $FF (= 0 bis 15*16+15 = 255) annehmen. Eine Adresse aus zwei Bytes umfaßt vier Hexstellen mit Werten von $0000 bis $FFFF (= 0 bis 15*4096+15*256+15*16+15*1 = 65535).

```
Umrechnung Hex nach Dezimal:
z.B. $7D95       =
        5  * 1
      +9   * 16
     +13   * 256
     +7    * 4096  = 32149 = DEC( "7D95" )  .
```

Umrechnung Dezimal nach Hex:
Zuerst ggf. wie oben in LO- und HI-Byte zerlegen. Innerhalb jedes Bytes BY Zerlegung in LO-Nibble LN und HI-Nibble HN:
HN = INT(BY / 16): LN = BY - 16 * HN
Umwandlung in Hexziffer: PRINT HEX$(Dezimalzahl)

Bei reinen Binärzahlen ist eine Stelle jeweils nur 2mal mehr wert als die rechts davon (Kennzeichnung durch vorangestelltes %). In BASIC können Zahlen, die bis zu 16stelligen Binärzahlen entsprechen, durch die Operatoren AND, OR und NOT binärverknüpft werden. Zur Wirkungsweise dieser Befehle muß immer die Binärdarstellung der beteiligten Zahlen klar sein:

```
Umrechnung Binär nach Dezimal:
z.B.    %1111 = 1*8 + 1*4 + 1*2 + 1*1 = 15
        %1001 = 8 + 1 = 9
Verknüpfung durch Operatoren:
z.B.    5 OR  7 = %101 OR  %111 = %111 = 7
        3 AND 6 = %011 AND %110 = %010 = 2
```

Bei der Negation NOT wird das 'Zweierkomplement' gebildet (bitweise Invertierung und Addition vonl):

```
        NOT  10 = NOT %0000 0000 0000 1010
                =     %1111 1111 1111 0101 + 1
                =     %1111 1111 1111 0110
                =     -11  (neg. Zahl wegen gesetztem höchsten Bit),
        NOT  -1 = 0        (BASIC-Werte für 'wahr' und 'falsch').
```

Einheit KByte: 1 KByte = 1024 Bytes = (2 hoch 10) Bytes
Das K ist groß zu schreiben, um es vom herkömmlichen k (für kilo, Faktor 1000) zu unterscheiden:
1 KByte = 1,024 kByte   und   64 KByte = 65,536 kByte

****
* SPEICHERBELEGUNG (MEMORY MAP) *
****

Adresse		RAM	ROM
$FFFF	65535		* ROM-BANK-HIGH *
$FFFE	65534		IRQ-Vektor
$FFFC	65532		RES-Vektor
$FFFA	65530		NMI-Vektor
			Kernal-Sprungtabelle
$FF81	65409		
		* I/O-Adressen (bis $FEFF 65279) und TED-CHIP-Register *	
$FD00	64768		
			ROM-BANKING-Routinen
$FC00	64512		
			Betriebssystem
$D800	55296		
			Character-Tabelle
$D000	53248		
			BASIC-Erweiterungen
$C000	49152		**********************
$BFFF	49151		
			* ROM-BANK-LOW *
			BASIC
$8000	32768		**********************
		****************************	
$3FFF	16383	* Ende 16-KByte-RAM *	
		BASIC-RAM (normal)	
		Bildschirmspeicher (Grafik)	
$2000	8192		
		Farbtabelle (Grafik)	
$1C00	7168		
		Luminanz (Grafik)	
$1800	6144		
$17FF	6143	Ende BASIC-RAM bei Grafik	
		BASIC-RAM	
$1000	4096		
		Bildschirmspeicher (Text)	
$0C00	3072		
		Farbspeicher (Text)	
$0800	2048		
		Systemspeicher	
$0000	0		

```
***************
* I N D E X *
***************
```

Abfrage-Anweisung (IF...THEN...ELSE) 155
AGELEITETE MATHEMATISCHE FUNKTIONEN 210
ABKÜRZUNGEN 202
ABS(X) Numerische Funktion 183
Absoluter Wert einer Zahl (ABS(X)) 183
Abspeichern s.u. Speichern
AC Akkumulator 224
Achteck zeichnen 143
Addition (+) 71,199
Alphanumerische Zeichenketten 195,222
AND Logischer Operator 156,200,227
Anführungszeichen-Modus 54,124,217
Anhalten des Programmablaufs (STOP) 178
Anhalten des Programmablaufs (WAIT) 181
ANSCHLUSS UND INBETRIEBNAHME s.u. TEIL 1 1
ANSCHLÜSSE UND SCHALTER 3
Antennen-Kabel s.u. RF-Anschlußbuchse 7
Antennenbuchse 7
Anweisungen 121,141
Anweisungsformat 122
ANWEISUNGEN, WEITERE INFORMATIONEN ZU DEN GRAPHIK- 182
ARBEITSSPEICHER UND BILDSCHIRM LÖSCHEN 60
Arcus Tangens (ATN(X)) 183
Are you sure? 42
Arithmetische Operatoren 199
ASC(X) Numerische Funktion 183
ASCII-Kode 215
ASCII-Kode als String (ASC(X$)) 183
ASCII-Kode -> String (CHR$(X)) 191
Assemble (Monitor) 218,219
ATN(X) Numerische Funktion 183
AUSFÜHRUNG VON RECHENOPERATIONEN 69
Ausgabe-Anweisung (PRINT) 165
Ausgabe-Anweisung (PRINT#) 166
Ausgabe Formatiert (PRINT USING) 167
Ausgabe (Print) umleiten (CMD) 144
Auslassungszeichen (...) 124
Ausmalen von Flächen s.u. Paint-Anweisung 103
AUTO-Kommando 125
Automatischer Programmstart (Diskette) 17
Automatische Zeilennumerierung (AUTO) 125

BACKUP-Kommando 125
BASIC 3.5, ANWEISUNGEN 141
BASIC 3.5, KOMMANDOS 125
BASIC 3.5 LEXIKON 119
BASIC 3.5 LEXIKON/EINLEITUNG 119
BASIC-OPERATOREN 199
BEDIENUNG DER TASTATUR 13
Bedienungshandbuch 2
BEFEHLE ZUM BILDSCHIRMAUSDRUCK, WEITERE 72
BEFEHLS-ABKÜRZUNGEN 202
BEFEHLS-EINGABE 55
Befehlskanal abfragen (DS,DS$) 198
Bemerkungen im Programm (REM) 173
BENUTZUNG DER SOFTWARE 31
Berechnungen (Vorrangordnung, Priorität) 71
BILDSCHIRM 48
BILDSCHIRM-FENSTER 61
Bildschirm löschen 20,60,176
Bildschirm-Kode 212
Bildschirm-Meldung (Einschalten) 9
Bildschirm-Randfarbe s.u. COLOR-Anweisung 88
BILDSCHIRM UND ARBEITSSPEICHER LÖSCHEN 60
BILDSCHIRMAUSDRUCK, WEITERE BEFEHLE ZUM 72
Bildschirmbereiche, Nummern der 89
Bindestrich (-) 124
Bit 226
Bit-mapped Bereich (HI-RES) 155
Bit-Map skalieren (SCALE) 175
Bogenmaß s.u. Sinus 188
BOX-Anweisung 98,141
Briggsscher Logarithmus (LOG Basis 10) 185,210
Brüche und Dezimalstellen 67
Buchstabenfarbe 25
Byte 226

C= (COMMODORE-Taste) 21,27
CARTRIDGES (STECKMODULE) 32
Chaining (Verketten) 129,135
CHAR-Anweisung 97,142
CHR$(X) String-Funktion 191
CIRCLE-Anweisung 101,143
CLOSE-Anweisung 144
CLR-Anweisung 144,225
CMD-Anweisung 144
COLOR-Anweisung 145
COLLECT-Kommando 126
COMMODORE 16 - die Musikmaschine 16
COMMODORE-Taste s.u. C= 21,27
Compare (Monitor) 218,220
CONT-Kommando 126
Control-Taste 20
Controlfunktinen darstellen 54,217
COPY-Kommando 127
COS(X) Cosinus, Numerische Funktion 183
Cursor-Position, Momentane (POS(X)) 193
Cursor-Sprung (SPC(X)) 193
CURSOR-Steuerung 18,217

DATA-Anweisung 146
Datassette 34
Datassetten-Anschluß 5
DATA-Werte 115,146
Datenaustausch (OPEN) 161
DEC Numerische Funktion 184,227
DEF FN-Anweisung 146
Definierte Funktion (DEF FN) 146
DELETE-Kommando 128
Dezimalstellen, Brüche und 67
Dezimalwert eines HEX-Strings (DEC) 184,227
DIM-Anweisung 147
Dimensionieren von Feldern (DIM) 147
Directory s.u. Inhaltsverzeichnis 45,128
DIRECTORY-Kommando 128
Direkt-Modus 55,120
Disassembler (Monitor) 218,220
DISK-FEHLERMELDUNGEN 206
DISKETTEN 39
Disketten bereinigen (COLLECT) 126
Disketten duplizieren (BACKUP) 125
Disketten formatieren (HEADER) 130
Diskettenfiles kopieren (COPY) 127
Diskettenlaufwerk 39
Diskettenprogramme laden (DLOAD) 40,129
Diskettenprogramme speichern (DSAVE) 44,130
Division (/) 71,199
DLOAD-Kommando 28,40,129
DO/LOOP/WHILE/UNTIL/EXIT-Anweisung 148
Dollar 196,226
DOS 206
DRAW-Anweisung 95,149,182
Drehung (BOX) 99,141
Drehwinkel (BOX) 99,141
Drehwinkel (CIRCLE) 101,143
Dreieck zeichnen 143
Druckzonen 1 bis 4 73
DS Reservierte Variable 198,206
DS$ Reservierte Variable 198,206
DSAVE-Kommando 28,130
Dump (M, Monitor) 223
Duration s.u. Klangdauer 109,111

Echtzeituhr-Variable (TI,TI$) 198
Eck-Koordinaten (BOX) 99,141
Editieren 19,57
Ein/Aus-Schalter 3,9
Eindimensionale Felder = Vektoren 197
Eingabe-Anweisung (GET) 151
Eingabe-Anweisung (GETKEY) 152
Eingabe-Anweisung (GET#) 153
Eingabe-Anweisung (INPUT) 157
Eingabe-Anweisung (INPUT#) 158
EINGABE VON BEFEHLEN 55
EINLEITUNG/TEIL 3 32
EINLEITUNG/TEIL 4 48
EINLEITUNG/TEIL 5 66
EINLEITUNG/TEIL 7 108
EINLEITUNG/BASIC 3.5 LEXIKON 120
Einzelpunktansteuerung s.u. Hochauflösende Grafik 93
Einzelzeichen-Abfrage (GET) 151
Einzelzeichen-Abfrage (GETKEY) 152
Einzelzeichen-Abfrage (GET#) 153
EL Reservierte Variable 198
Ellipse zeichnen 143
ELSE-Operator s.u. IF... 155
END-Anweisung 149
Entwerfen von Spielkarten 80
EOF (End OF File) 222
EOT (End Of Tape) 137
ER Reservierte Variable 198
ERR$(N) String-Funktion 191
Error Trapping s.u. Fehlersuchroutinen 198
Erste Programmierschritte 52
ERSTE SCHRITTE/TEIL 4 47
Esc-Taste 24
Escape s.u. Esc-Taste 24
Escape-Modus 62
Exclusiv-ODER Verknüpfung 181
EXIT-Anweisung s.u. DO/LOOP/... 148
EXP(X) Numerische Funktion 184
Exponentialfunktion (EXP) 184

Farbnummern (COLOR) 89,145
FARB-TASTEN 25
FARBÄNDERUNG UND REVERSE DARSTELLUNG 49
FARBEN UND GRAFIK/TEIL 6 79
FARBEN, STEUERUNG DER 88
Farbhelligkeit (Luminanz) 89
Farbhintergrund des Zeichens 95
Farbintensität der Farbzone (RLUM(N)) 186
Farbmonitor s.u. Video-Anschluß 6
Farbquelle, Farbzone der (RCLR(N)) 95,186
Farbregister-Nummer s.u. PAINT 163
Farbzone der Farbquelle (RCLR(N)) 186
Farbzone, Farbintensität der (RLUM(N)) 186
Farbzonen 89,141,145
Fehleranzeige, -kanal (Disk) 198,206
FEHLERBEHANDLUNG 56
Fehlerkorrektur 56
Fehlermeldungen (ERR$(ER)) 198,204,206
Fehlersuche abschalten (TROFF) 180
Fehlersuche einschalten (TRON) 180
FEHLERSUCHE, KLEINE 10
Fehlersuchroutinen-Variable (ER, EL) 198
Fehlerüberwachung im Programm (TRAP) 179
Fehlerzustand-String (ERR$) 191
FELDER / MATRIZEN 196
Felder dimensionieren (DIM) 147
FENSTER/BILDSCHIRM 61
Fernsehkabel s.u. RF-Anschlußbuchse 7
Feuerbefehl (Joystick) 185
File lesen 158,162
File öffnen (OPEN) 162
File schließen (CLOSE) 144
File schreiben 162,166
Filearten (Modi) 162
Filenamen umbenennen (RENAME) 136
Fill (Monitor) 218,221
Flächen malen (PAINT) 163
Flash/On- / Flash/Off-Taste 23
Fließkomma-Variable 76
Floating Point Akkumulator 188
Floppy Disk Drive 39
FNxx(X) Numerische Funktion 184
FOR...TO...STEP-Anweisung 150
Formatieren (Headern) einer Diskette 42,130
Formatierte Ausgabe (PRINT USING) 167
Formatmaske 122
Formatregeln 122
Formatvorgabe s.u. PRINT USING 167
Fortführen eines Programms (RESUME) 174
FRE(X) Freier Speicherbereich, Sonstige Funktion 193
Frequenz 110,211
FUNKTIONEN 183
Funktion, Benutzerdefinierte 78,146,184
FUNKTIONEN s.u. Numerische Funktionen 77,183
FUNKTIONEN s.u. String-Funktionen 191
FUNKTIONEN s.u. Sonstige Funktionen 193
FUNKTIONS-TASTEN 28,30,131

Ganzzahl-Anteil (INT(X)) 185
Ganzzahl-Variablen 195
Garantiekarte 2
Geräusche s.u. Tonerzeugung 110
Geräuscherzeugung (SOUND) 176
GET-Anweisung 151
Geteilter Bildschirm s.u. Grafik-Modus 92
GETKEY-Anweisung 152
GET#-Anweisung 153
Getrennter Speicherbereich s.u. Grafik-Modus 92
Gewittersturm 113
Gleitkomma-Variable (Fließkomma-) 196
Go (Monitor) 218,221
GOSUB-Anweisung 153
GOTO-Anweisung 154
Grafik-Anweisung 92
GRAFIK-ANWEISUNGEN, WEITERE INFORMATIONEN ZU DEN 182
GRAFIK, HOCHAUFLÖSENDE 93
GRAFIK, MEHRFARBIGE 103
Grafik-Modus 91
Grafik-Modus (GRAPHIC) 154
Grafik-Modus, Gegenwärtiger (RGR(X)) 186
GRAFIK-TASTEN 26,80
GRAFIK UND FARBEN/TEIL 6 79
GRAFIKZEICHEN 80
GRAPHIC-Anweisung 91,154
Groß-/Kleinbuchstaben 15,202,217
GRUNDAUSSTATTUNG 2
GRUNDRECHENFUNKTIONEN, ZAHLEN UND 66
Grundrechnungsarten 66

HEADER-Kommando 42,130
Headern s.u. Formatieren 42
Helligkeit der Farbe (Luminanz) 89
Helligkeitswert 145,186
HELP-Kommando 131
Help-Taste 30,56
Hexadezimaldarstellung 226
HEX$(N) String-Funktion 191,227
Hexadezimalwert einer ganzen Zahl (HEX$(N)) 191,227
HI-Byte 188,226
Hi-Res s.u. Hochauflösende Grafik 93
Hintergrund-Farbe s.u. COLOR-Anweisung 88
Hochauflösende Grafik s.u. Grafik-Modus 91,93
Home/Clear-Taste 19
Home-Position 19
Hunt (Monitor) 218,222

Identität (ID) der Diskette 42,130
IF...THEN (...ELSE)-Anweisung 155
INBETRIEBNAHME 7,9
INBETRIEBNAHME, ANSCHLUSS UND s.u. TEIL 1 1
Index-Zahlen s.u. Matrizen/Felder 196
Indirekt-Modus s.u. Programm-Modus 55
Individuelles Zeichnen (DRAW) 149
INFORMATIONEN ZU DEN GRAPHIK-ANWEISUNGEN, WEITERE 182
Inhaltsverzeichnis (DIRECTORY) der Diskette 45,128
INPUT-Anweisung 157
INPUT#-Anweisung 158
Inst/Del-Taste 18
Inst/Del-bei Korrekturen 57
INSTR Numerische Funktion 184
INT(X) Numerische Funktion 185
Integer-Variable 76
Interpunktionszeichen (Ausdruck) 72

JOY(N) Numerische Funktion, Joysticks 185
Joy 1 / Joy 2 s.u. Joystick-Anschlüsse 5
Joystick-Abfrage (JOY(n)) 185
Joystick-Anschlüsse 5

K (Kilo) 227
KASSETTEN 34
Kassetten-Programme (Laden) 34,36
KEY-Kommando 131
Keyword s.u. Schlüsselwort 122,202
Klammern (Eckige, Spitze) 123
Klammern (Runde) 71,124
Klang-Dauer (Duration) s.u. Tonerzeugung 109,111
Klang-Dauer s.u. SOUND 176
KLANGEFFEKTE ERZEUGEN 111
Klaviertastatur (Simulation) 114
KLEINE FEHLERSUCHE 10
Komma (,) 72,124,157,166
KOMMANDOS IN BASIC 3.5 125
KOMMANDO- UND ANWEISUNGSFORMAT 122
Kontroll- Lampe 'POWER' 3
Koordinaten (X,Y) 93,182
Koordinaten des Pixel-Cursors, Gegenwärtige (RDOT(N)) 186
Kopieren von Programmen (COPY) 127
Korrekturen 56
KREISE, VIELECKE UND MALEREI, QUADRATE 98
Kreise zeichnen 100,143
Kreissegment (CIRCLE) 143

Laden von bestimmten Kassetten-Programmen 36
Laden von Kassette oder Diskette (LOAD) 134
Laden von Disketten-Programmen 40
Laden von Kassetten-Programmen 34
Laden von Rechteckflächen (GSHAPE) 177
Laden von Steckmodul-Programmen 32
Länge eines Strings (LEN(X$)) 191
Lautstärke-Pegel (VOL) 109,180
LEFT$(X$,X) String-Funktion 191
LEN(X$) Numerische Funktion 191
LET-Anweisung 158
Leuchtreklame-Effekt 104
LEXIKON, BASIC 3.5 119
LINIEN UND ÜBERSCHRIFTEN, PUNKTE 95
Linien ziehen 96
Listen eines Programms (LIST-Kommando) 133
Liste von Elementen (DATA) 146
LOAD-Kommando 134
Load (Monitor) 218,222
LOCATE-Anweisung (Pixel-Cursor) 97,159,182
Löschen des Bildschirms 20,60,176
Löschen (Erase) einer Zeile 59
Löschen von Programmen (NEW) 135
Löschen von Diskettenfiles (SCRATCH) 135
Löschen von Programmzeilen (DELETE) 128
LO-Byte 188,226
LOG(X) Numerische Funktion 185
Logarithmus, Briggsscher (LOG Basis 10) 185
Logarithmus, Natürlicher (LOG(X)) 185
Logische Operatoren 200,227
LOOP s.u. Schleife 53
Luminanz s.u. Farbhelligkeit 89
Luminanz s.u. Helligkeitswert 145

Malen von Flächen (PAINT) 163
MALEREI, QUADRATE, KREISE, VIELECKE UND 98
Maschinenprogramm-Start s.a. USR(X) 188
Maschinenprogramm starten 179,221
Maschinensprache-MONITOR 159,218
Matrix s.u. Felder dimensionieren 147
MATRIZEN / FELDER 196
Mehrfachlesen von DATA-Zeilen (RESTORE) 174
MEHRFARBIGE GRAFIK 103
Mehrfarben-Befehl s.u. Mehrfarbige Grafik 103
Mehrfarben-Modus (Buchstaben/Zeichen) 142
Mehrfarbige Grafik s.u. Grafik-Modus 91,103
Memory-Dump (Monitor) 218,223
Memory-Expansion-Buchse s.u. Steckmodul-Anschluß 5
Memory-Map 228
MID$(X$,S,X) String-Funktion 192
Mittelpunkt-Koordinaten (CIRCLE) 101,143
Mnemonik 219,220
Modus, Anführungszeichen-/Quote- 57,217
Modus, Grafik- 91
Monitor-Anschluß 6
MONITOR-Anweisung s.u. Maschinensprache 159,218
Monitorbetrieb 9
Multiplikation (*) 71,199
MUSIK, TÖNE UND 107
Musik, Wir machen 114
Musikmaschine (COMMODORE 16) 116

NAMEN, VARIABLEN- 195
Natürlicher Logarithmus (LOG(X)) 185
Netzgerät 2
Netzgeräte-Anschluß 4,8
Neudefinieren von Zeichen (PUDEF) 172
NEW-Kommando 61,135
NEXT-Anweisung 160
Nibble 226
NOT Logischer Operator 156,200,227
Notation, Wissenschaftliche 68,195
Noten erzeugen 108
Notenwert s.u. Tonerzeugung 109
Notenwert s.u. SOUND 176,211
Numerische Funktionen 77,183
Numerische Variablen 195
Nummern, Farb- 89
Nummern-Raute (#) 123,167

ON-Anweisung 161
OPEN-Anweisung 161
OPERATOREN, VARIABLEN UND 195
OPERATOREN s.u. Arithmetische Operatoren 199
OPERATOREN s.u. Vergleichsoperatoren 200
OPERATOREN s.u. Logische Operatoren 200
OR Logischer Operator 156,200,227
OVERFLOW 169,205

PAINT-Anweisung 103,163
Parameter 123
PC s.u. Pixel-Cursor 141,182
PC (Program Counter) 224
PEEK(X) Numerische Funktion 186
PERIPHERIE 11
Peripherie-Anschluß 4
Pi Sonstige Funktion 193
Pi-Taste (π) 68
Pi Zahl (3.14159265) 193
Pixel s.u. Mehrfarbige Grafik 103
Pixel-Cursor 141,182
Pixel-Cursor-Koordinaten, Gegenwärtige (RDOT(N)) 186
Pixel-Cursor-Plazierung (LOCATE) 97,159
Platzhalter-Symbole (revers) 54,217
Plazierung des Pixel-Cursors (LOCATE) 97,159
POKE-Anweisung 164
POS(X) Sonstige Funktion 193
Position des Cursor, Momentane (POS(X)) 193
Potenzieren 71,184,199
Power-Buchse 8
PRINT-Anweisung 72,165
PRINT#-Anweisung 166
PRINT USING-Anweisung 167
Priorität bei Berechnungen (Klammern) 71,199
Programm-File (PRG) 162
Programm fortsetzen (RESUME) 174
Programm-Fortsetzung (CONT) 126
Programm-Modus (Indirekt-Modus) 55,120
Programm-Sprung (GOTO) 154
Programm-Start (RUN) 35,40,137
Programm-Stopp (END) 149
Programmablauf anhalten (STOP) 178
Programmablauf anhalten (WAIT) 181
Programmablauf unterbrechen 17,61
Programmzeilen-Nummer 133
Programmzeilen umnumerieren (RENUMBER) 136
PROGRAMMIERSCHRITTE/ERSTE 52
Programmzeile (40/80 Zeichen) 55,75
Programmzeilen löschen 59,128
Prompt 219
Prozent (%) 196,227
PUDEF-Anweisung 172
PUNKTE, LINIEN UND ÜBERSCHRIFTEN 95
Punktmatrix s.u. Hochauflösende Grafik 93

QUADRATE, KREISE, VIELECKE UND MALEREI 98
Quadratwurzel (SQR(X)) 188
Quote-Modus 57,217

Radius (CIRCLE) 101,143
Rauschen s.u. Geräusche 110,112
Raute zeichnen 143
RCLR(N) Numerische Funktion 186
RDOT(N) Numerische Funktion 186
READ-Anweisung 173
Ready-Modus 17
Rechenfunktion (FUNKTION) 183
RECHENOPERATIONEN, AUSFÜHRUNG VON 69
RECHENOPERATIONEN UND ZAHLEN/TEIL 5 65
Rechtecke zeichnen 98
Rechteckflächen abspeichern bzw. laden (SSHAPE/GSHAPE) 177
Register (Monitor) 218,224
Relationale Operatoren s.u. Vergleichsoperatoren 200
Relatives File (REL) 162
Relative Koordinaten 182
REM-Anweisung 173
RENAME-Kommando 136
RENUMBER-Kommando 136
Repeat-Funktion 18
RESERVIERTE VARIABLEN-NAMEN 197
Reset-Knopf/Taste 4,60
RESTORE-Anweisung 174
RESUME-Anweisung 174
RETURN-Anweisung 175
Return-Taste 16
Reverse Darstellung 22,54,212,217
Reverse-Flag 142
REVERSE DARSTELLUNG UND FARBÄNDERUNG 49
RF-Anschlußbuchse (f. TV-Kabel) 6
RGR(X) Numerische Funktion 186
RIGHT$(X$,X) String-Funktion 192
RLUM(N) Numerische Funktion 186
RND(X) Numerische Funktion 186
Rücksprung aus Subroutine (RETURN) 175
RUN-Kommando 137
Run/Stop-Taste 17
Rvs/On- / Rvs/Off-Taste 22

SAVE-Kommando 37,137
Save (Monitor) 218,224
SCALE-Anweisung 175
SCHALTER UND ANSCHLÜSSE 3
Schleife (Loop) 53
Schleifen (FOR...TO...STEP) 78,112,150
Schleifendurchlauf (NEXT) 160
Schleifen-Anweisungen (DO/LOOP/WHILE/UNTIL/EXIT) 148
Schließen eines Files (CLOSE) 144
Schlüsselwörter (Keywords) 55,122,197,202
Schreibmaschinen-Modus 15
SCNCLR-Anweisung 60,176
SCRATCH-Kommando 139
Scrollen (ESC) 63
Scrollen verlangsamen 22
Selbstdefinierte Funktionen 78,146,184
Semikolon (;) 72,165,124
Sequentielles File (SEQ) 162
Serial-Buchse s.u. Peripherie-Anschluß 4
SGN(X) Numerische Funktion 188
Shift-Taste 17
Simulationen per Zufallszahl 187
SIN(X) Numerische Funktion 188
Sinus-Funktion (SIN(X)) 188
Skalierung der Bit-Map (SCALE) 175
Software-Benutzung s.u. Teil 3 31
SONDER-TASTEN 16
Sonstige Funktionen 193
SOUND (TONERZEUGUNG) 109
SOUND-Anweisung 176
Soundregister 211
SP (Stackpointer) 224
Spalte 93,142
SPC(X) Sonstige Funktion 193
Speicheradresse-Flag 135
SPEICHERBELEGUNG 228
Speicherbereich-Abfrage, Freier (FRE(X)) 193
Speicherbereich-Zugriff (POKE) s.a. PEEK(X) 164
Speicherinhalt(PEEK(X)) 186
Speichern auf Kassette oder Diskette (SAVE) 137
Speichern von Programmen auf Diskette 44,130
Speichern von Programmen auf Kassette 37,137
Speichern von Rechteckflächen (SSHAPE) 177
Speicherstellen 1281/1282 s.a. USR(X) 188
Spielkarten/Wie man ..entwirft 80
Spielkartensymbole 80
SQR(X) Numerische Funktion 188
SSHAPE/GSHAPE-Anweisung 177

ST Reservierte Variable 197
Stackpointer 224,225
Standardwerte für Funktionstasten 132
Startadresse für Unterprogramm (USR(X)) 188
Starten eines Programms (RUN) 35,40,137
Starten eines Maschinen-Programms 179,221
Status-Register 224
Status-Variable (ST) 197
STECKMODULE (CARTRIDGES) 32
Steckmodul-Anschluß 5
Steckmodul-Programme (Laden) 32
Stellenschreibweise 226
STEP-Anweisung s.u. FOR... 112,150
STEUERUNG DER FARBEN 88
Stimme s.u. SOUND 176
Stimmen-Nummer s.u. Tonerzeugung 109
STOP-Anweisung 178
Stop/Eject-Taste (Datassette) 36
Stoppen des Programmablaufs (STOP) 178
STR$(X) String-Funktion 192
String-Äquivalent einer Zahl (STR$(X)) 192
String aus ASCII-Kode (CHR$(X)) 191
String des Fehlerzustandes (ERR$) 191
String-Funktionen 191
String Links (LEFT$(X$,X)) 191
String Mitte (MID$(X$,X,Y)) 192
String Rechts (RIGHT$(X$,X)) 192
String-Variablen 195
Stringlänge (LEN(X$)) 191
Stringumwandlung in Zahl (VAL(X$)) 189
Stringvergleich (INSTRING) 184
Subroutine (GOSUB) 153
Subtraktion (-) 199
Syntax von Basic 3.5 120
SYS-Anweisung 179
Systemuhr s.u. Echtzeit-Variable (TI,TI$) 198

TAB(X) Sonstige Funktion 193
TAB-ähnlicher Cursor-Sprung (SPC(X)) 193
TAB-Funktion 87
TAB-Funktion (Spaltensprung) (TAB(X)) 193
TAN(X) Numerische Funktion 188
Tangens-Funktion (TAN(X)) 188
TASTATUR (BEDIENUNG) 13
TASTENFELD 14
TEDMON 218
TEIL 1 ANSCHLUSS UND INBETRIEBNAHME 1
TEIL 2 BEDIENUNG DER TASTATUR 13
TEIL 3 BENUTZUNG DER SOFTWARE 31
TEIL 4 ERSTE SCHRITTE 47
TEIL 5 ZAHLEN UND RECHENOPERATIONEN 65
TEIL 6 GRAFIK UND FARBEN 79
TEIL 7 TÖNE UND MUSIK 107
Text-Modus s.u. Grafik-Modus 91
THEN-Anweisung s.u. IF... 155
TI Reservierte Variable 198
TI$ Reservierte Variable 198
Tippfehlerbehebung 56
TÖNE UND MUSIK/TEIL 7 107
TONERZEUGUNG (SOUND) 109,176
Tonfrequenzen s.u. Tonerzeugung 110
Tongenerator 1 & 2 109
Tonhöhe s.u. Notenwert 110,211
Tonhöhe s.u. SOUND 176
Transfer (Monitor) 218,225
TRAP-Anweisung 179
Trennstrich (|) 124
TRICK (ZEICHENBEWEGUNG) 84
TRON-Anweisung 180
TROFF-Anweisung 180
TV-Kabel s.u. RF-Anschlußbuchse 6

Überprüfen von Programmen (VERIFY) 139
ÜBERSCHRIFTEN, PUNKTE, LINIEN UND 95
Uhrzeit s.u. Systemuhr 198
Umbenennen von Filenamen (RENAME) 136
Umnumerieren von Programmzeilen (RENUMBER) 136
Umwandlung String -> Zahl (VAL(X$)) 189
USR(X) Numerische Funktion 188
User-File (USR) 162

VAL(X$) Numerische Funktion 189
VARIABLE 75,195
Variablen auf Null setzen (CLR) 144
VARIABLEN-NAMEN 196
VARIABLEN-NAMEN, RESERVIERTE 197
VARIABLEN UND OPERATOREN 195
Variablen-Liste lesen (READ) 173
Variablen-Typen 76,124
Vektoren = eindimensionale Felder 197
Vergleichsoperatoren 200
VERIFY-Kommando 139
Verify (Monitor) 218,225
Verketten (Chaining) 129,135
Verknüpfung Exclusiv-ODER 181
Video-Buchse 6
Videokabel 6
VIELECKE UND MALEREI, QUADRATE, KREISE 98
VOL-Anweisung 109,180
VOLUMEN (LAUTSTÄRKEREGELUNG) 109
Vordergrundfarbe 95
Vorrangordnung bei Berechnungen 71
Vorzeichen (Mathematik) 70
Vorzeichen einer Zahl ermitteln (SGN(X)) 188

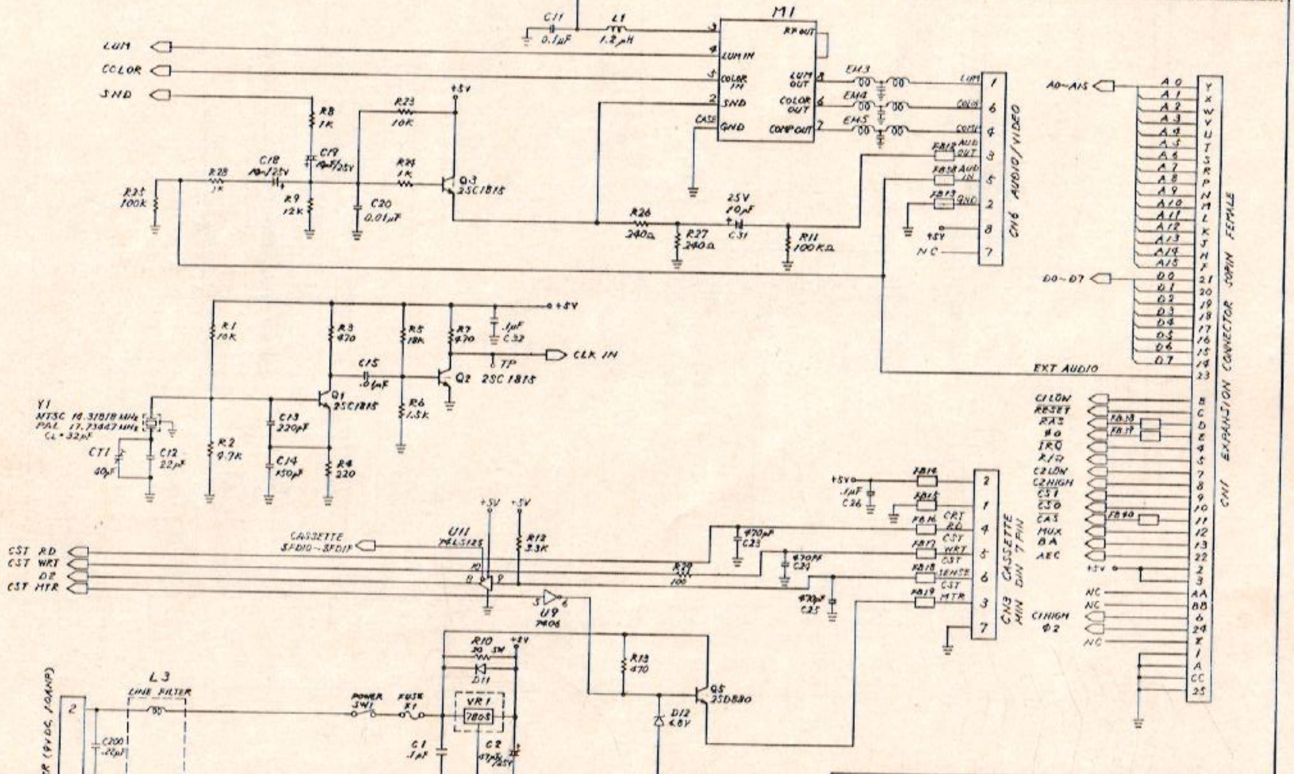
WAIT-Anweisung 181
Weißes Rauschen s.u. Tonerzeugung 110
WEITERE BEFEHLE ZUM BILDSCHIRMAUSDRUCK 72
WEITERE INFORMATIONEN ZU DEN GRAFIK-ANWEISUNGEN 182
Windows s.u. Bildschirm-Fenster 61
Winkel 99,101,141,143,182
Winkel (Bogenmaß) 188
Wissenschaftliche Notation 68,195
Würfelspiel s.u. Zufallszahl 187

X,Y Koordinaten 93,182
X-Register 224

Y-Register 224

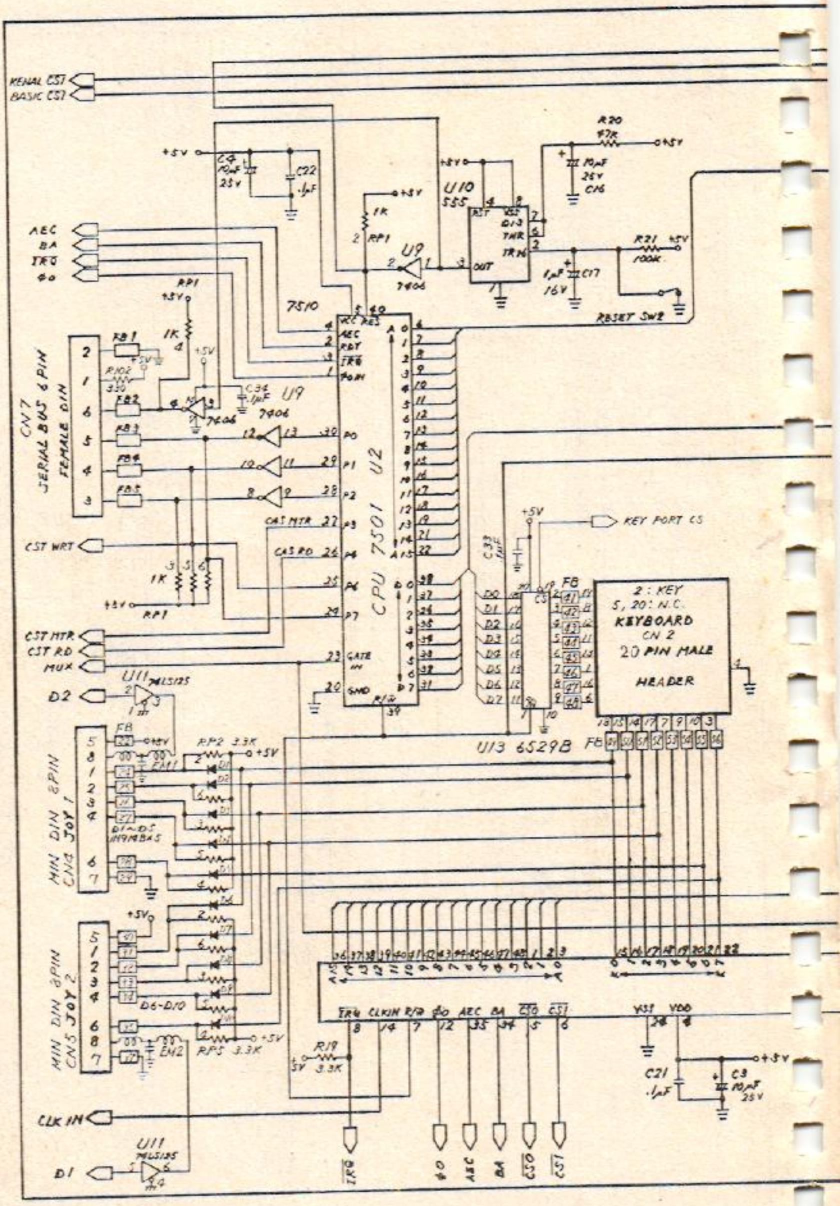
Zahl aus String (VAL(X$)) 189
Zahl Pi (3.14159265) 68,193
ZAHLEN UND GRUNDRECHENFUNKTIONEN 66
ZAHLEN UND RECHENOPERATIONEN/TEIL 5 65
Zeichnen, Individuelles (DRAW) 149
Zeichnen (Kreise) 100
Zeichnen (Rechtecke) 98
Zeichen einfügen s.u. Inst/Del Taste 18,57
Zeichen entfernen s.u. Inst/Del Taste 18,57
Zeichen neu definieren (PUDEF) 172
ZEICHENBEWEGUNG (TRICK) 84
Zeichnungsmodus s.u. Grafikmodus 91
Zeilennummern 52
Zeilennumerierung (AUTOmatische) 125
Zufallszahl (RND(X)) 186
Zusatz-Parameter s.u. Klammern 123
Zuweisung von Variablen (LET) 158

REVISIONS				
LTR	ZONE	DESCRIPTION	DATE	APPROVED
A		PRODUCTION RELEASE	3-9-84	W Pate

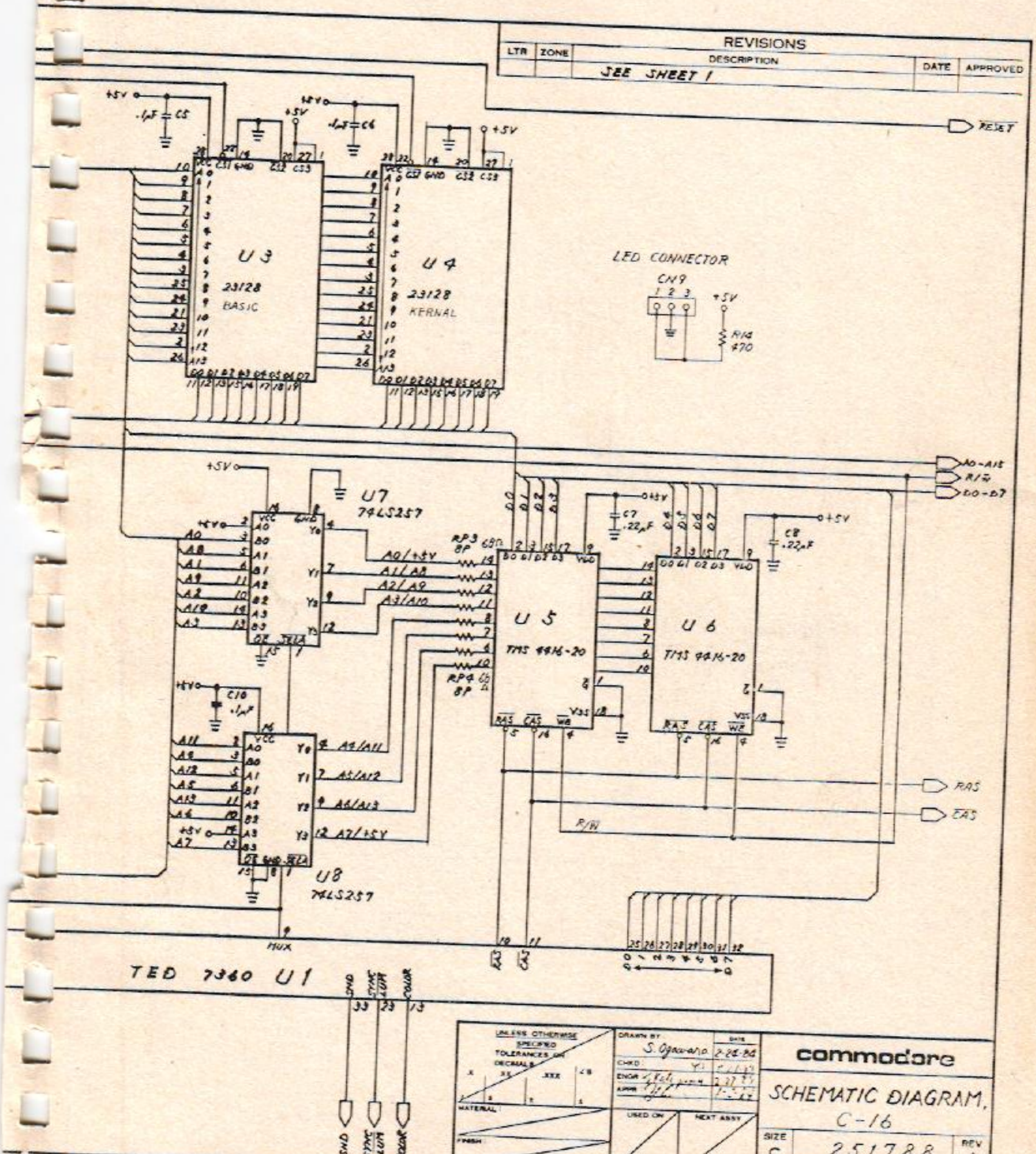


UNLESS OTHERWISE SPECIFIED TOLERANCES ON DECIMALS	DRAWN BY S. Ogawara	2-29-84	commodore
MATERIAL	USED ON C-16	NEXT ASSY 250433	SCHEMATIC DIAGRAM, C-16
FINISH			SIZE C — 251788 — REV A
			SCALE NONE — SHEET 1 OF 3

THIRD ANGLE SYSTEM — DIMENSION IN

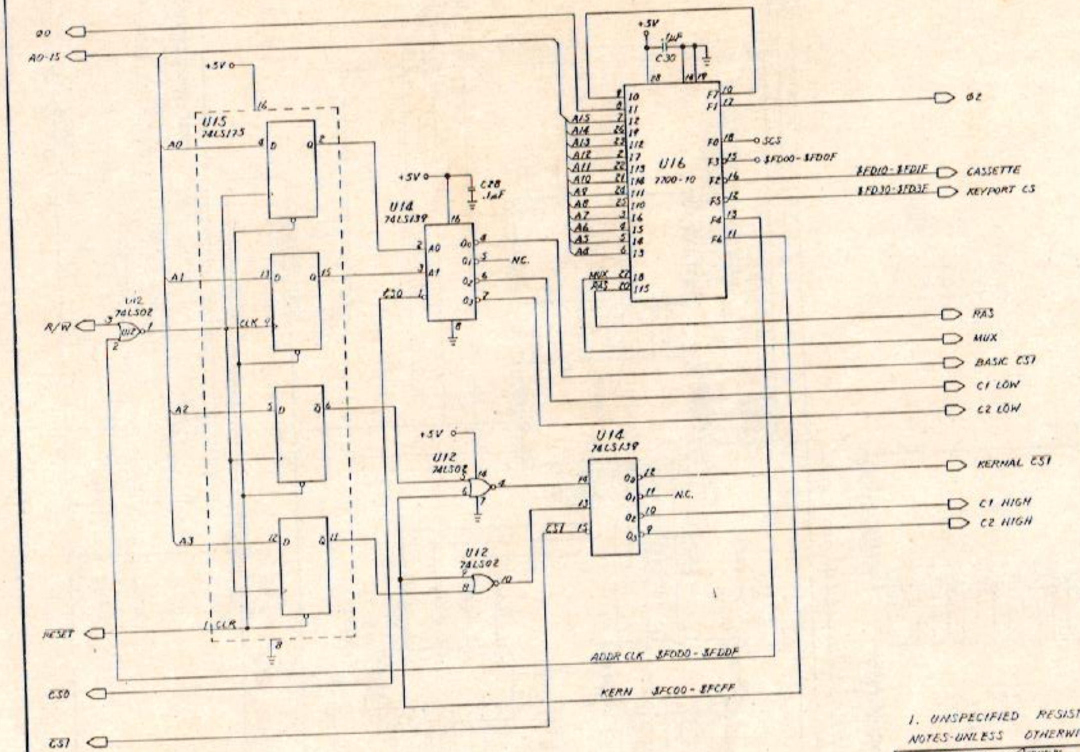


KENAL CST
BASIC CST
AEC
BA
IRQ
Φ0
CN7
SERIAL BUS 6PIN
FEMALE DIN
CST WRT
CST MTR
CST RD
MUX
D2
MIN DIN 8PIN
CN4 JOY 1
MIN DIN 8PIN
CN5 JOY 2
CLK IN
D1
U11 74LS125
U9 7406
U10 555
R20 47K
R21 100K
C16
C17
RESET SW2
7510
CPU 7501 U2
CAS MTR
CAS RD
GATE IN
GND
R/W
KEY PORT CS
U13 6529B
2 : KEY
5, 20 : N.C.
KEYBOARD
CN 2
20 PIN MALE
HEADER
D1~D5 1N914BAS
D6-D10
RP2 3.3K
RP5 3.3K
R19 3.3K
C21 .1μF
C3 10μF 25V
C22 .1μF
C4 10μF 25V
TRQ
CLKIN
R/W
AEC
BA
CS0
CS1
VSS
VDD



REVISIONS
LTR
ZONE
DESCRIPTION
DATE
APPROVED
SEE SHEET 1
RESET
U3
23128
BASIC
U4
23128
KERNAL
LED CONNECTOR
CN9
+5V
R14
470
C5
C6
U7
74LS257
C7
.22μF
C8
.22μF
C10
RP3
RP4
U5
TMS 4416-20
U6
TMS 4416-20
U8
74LS257
A0/+5V
A1/A8
A2/A9
A3/A10
A4/A11
A5/A12
A6/A13
A7/+5V
A0-A15
R/W
D0-D7
RAS
CAS
MUX
TED 7360 U1
SND
SYNC LUM
COLOR
UNLESS OTHERWISE SPECIFIED
TOLERANCES ON DECIMALS
MATERIAL
FINISH
DRAWN BY
S. Ogasawara
2-28-84
CHKD
ENGR
APPR
USED ON
NEXT ASSY
commodore
SCHEMATIC DIAGRAM, C-16
SIZE C
251788
REV A
SCALE NONE
SHEET 2 OF 3
THIRD ANGLE SYSTEM

LTR	ZONE	REVISIONS DESCRIPTION	DATE	APPROVED
		SEE SHEET 1		



NOTES-UNLESS OTHERWISE SPECIFIED:

1. UNSPECIFIED RESISTORS ARE 1/4 WATT ± 5%.

commodore

SCHEMATIC DIAGRAM, C-16

SIZE C — 251788 — REV A

SCALE NONE — SHEET 3 OF 3

DRAWN BY S Opasana

Commodore

Commodore GmbH
Lyoner Straße 38
D-6000 Frankfurt/M. 71

Commodore AG
Aeschenvorstadt 57
CH-4010 Basel

Commodore GmbH
Kinskygasse 40–44
A-1232 Wien

© Commodore Büromaschinen GmbH
Nachdruck, auch auszugsweise, nur mit schriftlicher Genehmigung von COMMODORE.

P/N 324980-02
Artikel-Nr. 580016

Änderungen vorbehalten